Goal-directed Proof Theory

Dov M. Gabbay
Department of Computer Science
King's College
London, UK [1]

Nicola Olivetti
Dipartimento di Informatica
Universitá di Torino
torino, Italy [2]

5th-Draft - April 1999

[1] *address:* Strand, London, WC2R 2LS, email: dg@dcs.kcl.ac.uk
[2] *address:* C.so Svizzera 185, 10149 Torino, Italy, email: olivetti@di.unito.it

Contents

Chapter 1

Introduction

1.1 Introduction

This book is about goal directed proof theoretical formulations of non-classical logics. It evolved from a response to the existence of two camps in the applied logic (computer science/artificial intelligence) community. There are those members who believe that the new non-classical logics are the most important ones for applications and that classical logic itself is now no longer the main workhorse of applied logic and there are those who maintain that classical logic is the only logic worth considering and that within classical logic the Horn clause fragment is the most important one.

The book presents a uniform Prolog-like formulation of the landscape of classical and non-classical logics, done in such a way that the distinctions and movements from one logic to another seem simple and natural; and within it classical logic becomes just one among many. This should please the non-classical logic camp. It will also please the classical logic camp since the goal directed formulation makes it all look like an algorithmic extension of Logic Programming. The approach also seems to provide very good compuational complexity bounds across its landscape.

The spectacular rise in non-classical logic and its role in computer science and artificial intelligence was fuelled by the fact that more and more computational 'devices' were needed to help the human at his work and satisfy his needs. To make such a 'device' more effective in an application area, a logical model of the main feature of human behaviour in that area was needed. Thus logical analysis of human behaviour became part of applied computer science. Such study and analysis of human activity is not new. Philosophers and pure logicians have also been modelling such behaviours, and in fact, have already produced many of the non-classical logics used in computer science. Typical examples are modal and temporal logics. They have been applied extensively in philosophy and linguistics as well as in computer science and artificial intelligence.

The landscape of non-classical logics applications in computer science and artificial intelligence is very wide and varied. Modal and Temporal logics have been profitably applied to verification and specification of concurrent systems [Manna and Pnueli 81], [Pnueli 81]. In the area of Artificial Intelligence as well as of distributed systems, the problem of reasoning about knowledge, belief and action has received much attention; modal logics [Turner 85], [Halpern and Moses 90] have been seen to provide the formal language to represent this type of reasoning. Relevance logics have been applied in natural language understanding and database updating [Martins and Shapiro 88]. ' Lambek's logic [Lambek 58] and its extensions are currently used for natural language processing. Another source of interest in non-classical logics, and in particular in so-called substructural logics (with the prominent case of *linear logic* [Girard 87]) has originated from the functional interpretation provided by the Curry-Howard's isomorphism between formulas and types in functional languages [Gabbay and DeQueiroz 92],[Wansing 90].

In parallel with the theoretical study of the logics mentioned above and their applications, there has been a considerable amount of work on their automation. The area of non-classical theorem proving is growing very rapidly, although it is yet not as developed as classical theorem proving.

Although there is a wide variety of logics, we can group the existing methodologies for automated deduction in few categories. Most of the ideas and methods for non-classical deduction have been derived from their classical counterpart. We have analytic methods such as tableaux, systems based on Gentzen's calculi, methods which extend and reformulate classical resolution, translation based methods, and goal-directed methods.

We can roughly distinguish two paradigm of deduction: *human-oriented* proof-methods versus *machine-oriented* proof methods. We can call a deduction method *human-oriented* if in principle, the formal deduction follows closely the way a human does it. In other words, we can understand how a deduction goes on and, more precisely, how each intermediate step is related to the original deductive query. With *machine-oriented* proof methods this requirement is not mandatory: the original deductive task might be translated and encoded even in another formalism. The intermediate steps of a computation might have no directly visible relationship with the original problem.

According to this distinction, natural deduction, tableaux and Gentzen calculi are examples of the human-oriented paradigm, whereas resolution and translation based methods are better seen as examples of the machine oriented paradigm. In particular resolution methods require us to transform the question "does Q follow from Δ?", into the question "is $\Delta^* \cup \{Q^*\}$ consistent?", where Δ^* and Q^* are preprocessed *normal forms* of Δ and Q. Stepping from Δ and Q to Δ^* and Q^* one may loose information involved in the original Δ and Q. Moreover, the normal forms may be natural only from the machine implementation point of view, and not supported by human way of reasoning.

Machine oriented methods are more promising from the point of view of efficiency than human-oriented proof methods. After all, efficiency, uniformity and reduction of the search space was the main motivation behind the introduction of resolution.

The basic features of goal-directed methods are that

- they are *human oriented*;
- they are a generalization of logic programming style of deduction.

Goal-directed methods can be seen as an attempt to fill the gap between the two paradigms, on one hand, they maintain the perspicuity of human-oriented proof methods, on the other hand, are not too far from an efficient implementation. There is another reason of interest in goal-directed proof search. Although we generally speak about deduction, there are namely several different tasks/problems which can be qualified as deductive. These different tasks might be theoretically reducible one to each other, but a method or an algorithm to solve one does not necessarily apply well to another. To make this point more concrete, assume we are dealing with a given logical system $\mathcal{L}$ (say classical, or intutionistic logic, or modal logic S4), we use the symbol $\vdash$ to mean both theoremhood and consequence relation in that logic. Compare the following problems:

1. given a formula A we want to know whether $\vdash A$, that is whether A is a theorem of $\mathcal{L}$.
2. We are given a set Γ containing 100.000 formulas and a formula A and we want to know whether $\Gamma \vdash A$.

3. We are given a set of formulas Γ and we are asked to generate all atomic propositions which are entailed by Γ.

4. We are given a formula A and a set of formulas Γ sucht that $\Gamma \not\vdash A$. We are asked to find a set of atomic propositions S such that $\Gamma \cup S \vdash A$.

The list of problems tasks might continue. We call the first problem 'Theorem-proving'. The second problem is close to deductive-database query answering. The third problem/task may occur when we want to revise a knowledge-base or a state description as an effect of some new incoming information. The fourth problem is involved in abductive reasoning and in practice one would impose various constraints on possible solution sets S. It is not difficult to see that the second, third and fourth problem are reducible to the first one. An algorithm to determine theoremhood can be used to solve the other problems as well. Suppose we have an efficient theorem prover P. Problem 2. can be reduced to check the theorem $\bigwedge \Gamma \rightarrow A$. Thus, we can feed P with this huge formula, run it and get an answer. However it might be that the formulas of Γ have a particular simple format and most importantly only a very small subset of them (say 10 formulas) are relevant to get the proof of A. Even if our theorem prover P has an optimal complexity in the size of the data $(\Gamma + A)$, we would rather prefer a deduction method which is in princible capable of concentrating on the data in Γ which are relevant to the proof of A and ignore the rest.

The theorem prover can be used to solve also Problem 3: just enumerate all atomic formulas p_i, check whether $\bigwedge \Gamma \rightarrow p_i$, give as output the atomic formulas for which the answer is yes. It is very likely that there are better methods to accomplish this task! For instance in the case that Γ is a set of Horn clauses, one can use a bottom-up evaluation; more generally, one would try to incrementally generate this set by a saturation procedure.

The theorem prover can be used to solve Problem 4 in a non deterministic way: guess a set S and check by the theorem prover whether $\Gamma \wedge \bigwedge S \rightarrow A$. Again, no matter how it is efficient our theorem-prover, it is obvious that there are better methods of performing this task, for instance one attempts a proof of A from Γ and determine as far as the proof proceeds what should be added to Γ (i.e. a solution S) to make the proof succeed. To perform this task one would prefer a method by which the extraction of such solution sets S from derivations is easy.

All these considerations, shows that the theorem-proving perspective is not the only possible way of looking at deduction. Another well-known example is proof-search: one might be interested not only in determining whether a formula is a theorem, but to find out a proof of the formula with certain features (this interest is intrinsic type-inhabitation problems).

The goal-directed paradigm we follow in this work is particularly well suited for deduction which involves a great amount of data (that is what we have qualified as deductive database perspective). Moreover goal-directed deductive procedures can be used to design abductive reasoning procedures [Eshghi89], although we will not develop this point further in the present work.

The goal-directed paradigm we adopt is the same as the one underlying logic programming. The deduction process can be described as follows: we have a structured collection of formulas (called a database) Δ and a a goal formula A, and we want to know whether A follows from Δ or not, in a specific logic. Let us denote by

$$\Delta \vdash^? A$$

the query "does A follows from Δ?" (in a given logic). The deduction is *goal-directed* in the sense that the next step in a proof is determined essentially by the form of the current goal: the goal is stewpwise decomposed, according to its logical structure, until we reach its atomic constituents. An atomic goal q

is then matched with the "head" of a formula $G' \to q$ (if any, otherwise we fail) in the database, and its "body" G' is asked in turn. This is what happens with the logic programming approach to Horn-clause computation in intuitionistic or classical logic, namely a Horn-clause can be read as a procedure

$$\Delta, a_1 \wedge a_2 \to c \quad \vdash^? \; c$$

reduces to

$$\Delta, a_1 \wedge a_2 \to c \quad \vdash^? \; a_1 \wedge a_2$$

A call to c reduce to a call to a_1 and to a_2. This procedural way of looking at clauses is equivalent to the declarative way namely to $\vdash$ provability (which for Horn-clauses coincides for classical and for intuitionistic logics). We ask ourselves, can we extend this backward reasoning, goal directed paradigm to all of classical and neighbouring logics? In other words, can we have a logic programming like proof system presentation for classical, intuitionistic, relevance and other logics? In this book we try to give a positive answer to this question.

As we will see, in order to provide a goal-oriented presentation of a large family of non-classical logics, we will have to refine this simple model of deduction in two directions: in a few words

- we may put constraints on use and "visibility" of database formulas, so that not necessarily all of them are available to match an atomic constituent.
- we may allow to re-ask a goal previously occurred in the deduction.

In the next section we will show a variety of examples of goal directed computations.

The concept of goal directed computation we adopt can also be seen as a generalization of the notion of *uniform proof* as introduced in [Miller et al. 91]. As far as we know, a goal-directed presentation have been given of (fragments of) intuitionistic logic [Gabbay and Reyle 84],[Miller 89], [McCarty 88a], [McCarty 88b], higher order logic [Miller et al. 91] and of some substructural logics, namely linear logic [Hodas and Miller 91], [Harland and Pym 91], relevant logic [Bollen 91]. In [Gabbay 92],[Gabbay and Kriwaczek 91], goal-directed procedures for classical and some intermediate logics are presented. In most of the literature, goal-directed procedures have been investigated for some specific logics but always thinking of them as a refinement of pre-existing deduction methods. This is for instance the case of the *uniform proof paradigm* promoted by Miller and others. In the uniform-proof framework, goal-directed proofs are proofs in a given Gentzen system (for a specific logic), which satisfy some additional constraint. It is the analysis of provability within the Gentzen formulation which allows one to identify what fragment of a logic, if any, admit a goal-directed procedure. In this sense the goal-directed proof procedure is a refinement of Gentzen formulation, it gives a discipline on how to find a proof in a given Gentzen calculus. A uniform proof system is called "an abstract logic programming" [Miller et al. 91]. The essence of a uniform-proof systems, or as we call it, a goal-directed system is that the proof-search is driven by the goal and that the connectives can be interpreted directly as search insturctions.

However, we do not have to necessaryly rely on a Gentzen calculus for developing a goal-directed system. It may even happen for specific systems that a proper formalization of provability by means of a Gentzen calculus is not known, or if known is not natural. Still a goal-directed formulation might be easily obtainable. We think that Gentzen calculi and goal-directed proof-metlods are related, but distinct concepts. We will not attempt at the beginning to provide a general definition of goal-directedness, (see next section for examples). It would be rather artificial, as to give a definition of "Tableaux" procedures, or Gentzen calculus. We will try to develop a uniform family of calculi for several types of logics. To this purpose, we consider mainly a minimal fragment of the logics we study, (namely the implicational fragment); this will make our presentation uniform, and will allows us to compare logics through their goal-directed procedure.

We develop goal-directed procedures for a variety of logics, stretching from modal, to relevance and intermediate logics. By means of the goal-directed formulation, one can prove cut elimination and other properties. ¿From a practical point of view, a goal-directed formulation of a logic may be used to design efficient, Prolog-like, deductive procedures (and even abductive procedures) for each logic. In this work we will mainly concentrate on implicational logics. One reason is the uniformity of treatment of all logics, the other is that as we regard implication as the basic connective of most logics. The prominence we give to implication is justified by the connection between the consequence relation of a logic (in the case it is defined) and its implication connective, through the deduction theorem. If a logic **L** contains an implication connective, then it is always possible to define a form of consequence relation $\vdash_{\mathbf{L}}$ by letting

$$A_1, \ldots, A_n \vdash_{\mathbf{L}} B \;\Leftrightarrow\; A_1 \to (A_2 \to \ldots \to (A_n \to B)\ldots) \text{ is valid in } \mathbf{L}.$$

For instance, we will define as above a form of consequence relation for modal logics, where the connective $\to$ is read as strict implication.

A further motivation to restrict our investigation, (at least at a first stage), to implicational logics, is the observation that most logics differ on their implication connective, whereas they coincide in the treatment of the other connectives.

1.2 A survey of goal-directed methods

To explain what we mean by goal-directed deduction style, we begin by recalling standard propositional Horn deductions. This type of deduction is usually interpreted in terms of classical resolution, but it is not the only possible interpretation [1] The data are represented by a set of propositional Horn clauses, which we write as

$$a_1 \wedge \ldots \wedge a_n \to b.$$

The a_i are just propositional variables and $n \geq 0$. In case $n = 0$, the formula reduces to b. This formula is equivalent to:

$$\neg a_1 \vee \ldots \vee \neg a_n \vee b.$$

Let Δ be a set of such formulas, we can give a calculus to derive formulas, called " goals" of the form $b_1 \wedge \ldots \wedge b_m$. The rules are something of the form:

- $\Delta \vdash^? b$ succeeds if $b \in \Delta$;
- $\Delta \vdash^? A \wedge B$ is reduced to $\Delta \vdash^? A$ and $\Delta \vdash^? B$;
-$\Delta \vdash^? q$ is reduced to
$\Delta \vdash^? a_1 \wedge \ldots \wedge a_n$, if there is a clause in Δ of the form $a_1 \wedge \ldots \wedge a_n \to q$.

The main difference from the traditional logic programming convention is that in the latter conjuction is eliminated and a goal is kept as a sequence of atoms $b_1, \ldots, b_m$. The computation does not split because of conjunction, all the subgoals b_i are kept in parallel, and when some b_i succeeds (that is $b_i \in \Delta$) it is deleted from the sequence. To obtain a real algorithm we should specify in what order we scan the database when we search for a clause whose head matches the goal. Let us see an example.

[1]For a survey on foundation of logic programming, we refer to [Lloyd 84] and to [Gallier 87].

Example 1.2.1 Let Δ contain the following clauses

1. $a \wedge b \rightarrow g$,
2. $t \rightarrow g$,
3. $p \wedge q \rightarrow t$,
4. $h \rightarrow q$,
5. $c \rightarrow d$,
6. $c \wedge f \rightarrow a$,
7. $d \wedge a \rightarrow b$,
8. $a \rightarrow p$,
9. $f \wedge t \rightarrow h$,
10. c,
11. f.

A derivation of g from Δ can be displyed in the form of a tree and it is displayed in Figure 1.1. The number in front of every non-leaf node indicates the clause of Δ which is used to reduce the atomic goal in that node.

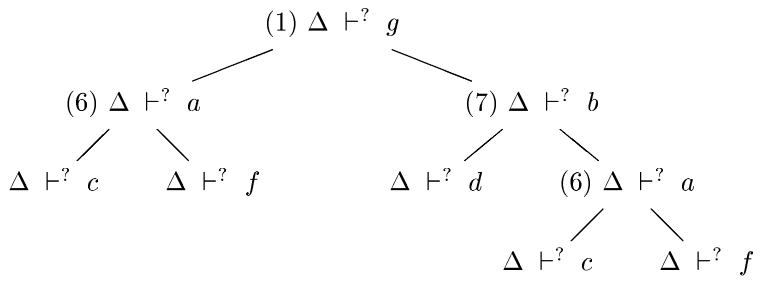


Figure 1.1:

■

We can make a few observations. First, we do not need to consider the whole database, it might be even infinite, and the derivation would be exactly the same; irrelevent clauses, those whose "head" do not match with the current goal are ignored. The derivation is driven by the goal, in the sense that each step in the proof simply replaces the current goal with the next one.

Notice also that in this specific case there is no other way to prove the goal, and the sequence of steps is entirely determined.

Two weak points of the method can also be noticed. Suppose that when asking for g we use the second formula, then we continue asking for t, then asking for h, and then we ask for t again. In other words, we are in a loop. An even simpler situation is the following

$p \rightarrow p \vdash^? p$,

We can go on asking for p without realizing that we are in a loop. To deal with this problem we should add a mechanism which ensures termination.

Another problem which bears on the efficiency of the procedure is that a derivation may contain redundant subtrees. This happens if the same goal is asked several times. In the previous example it happens with the subgoal a. In this case, the global derivation contains multiple subderivations of the same goal. It would be better to be able to remember whether a goal has already been asked (and succeeded) in order to avoid the duplication of its derivation. Whereas the problem of termination is crucial in the evaluation of the method (if we are interested in getting an anwer eventually), the problem of redundancy will not be considered in this work. However, avoiding the type of redundancy we have described has a dramatic effect on the efficiency of the procedure, for redundant derivations may grow exponentially in the size of the data.

Although the goal directed procedure does not necessarily produce the shortest proofs, nor does it always terminate, still it has the advantage that proofs, when they exist, are easily found. Making precise this notion of "easyness" in terms of proof-search space is admittedly rather difficult and we will not try do it here[2].

Let us see if we can extend this goal-directed approach to a broader fragment. We still consider the database as before but we allow to ask also clauses as goals. How we eveluate the following goal?

$$\Delta \ \vdash^? \ a_1 \wedge \ldots \wedge a_n \rightarrow b$$

This can be read as an hypothetical query. We can think of using the deduction theorem as a deduction rule:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B}$$

The above query is hence reduced to

$$\Delta \cup \{a_1, \ldots, a_n)\} \ \vdash^? \ b$$

This step can be also justified in the traditional refutational interpretation of deduction. To show that

$$\Delta \cup \{\neg(a_1 \wedge \ldots \wedge a_n \rightarrow b)\}$$

is not satisfiable, means to show that $\Delta \cup \{a_1, \ldots, a_n, \neg b)\}$ is unsatisfiable. If we are capable of treating clauses as goals, why don't we allow clauses as bodies of other clauses? We are therefore lead to consider a hypothetical extension of Horn deductive mechanism. This means that we can handle arbitrary hypothetical goals through the rule:

from $\Delta \ \vdash^? \ A \rightarrow B$, step to
$\Delta \cup \{A\} \ \vdash^? \ B$.

This straightforward extension of Horn logic captures exactly intuitionistic provability for this fragment, but not classical provability. In other words, the embedded implication behaves as intutionistic implication, but not as classical implication. This kind of implicational extension based on intuitionistic logic is part of many extensions of logic programming we have recalled above, ([Gabbay and Reyle 84],[Miller 89], [McCarty 88a], [McCarty 88b]).

But we can do more, suppose we label data to keep track of the way it is used in derivations. One may want to put some control on the way the data is used. Now the database is a labelled set of formulas $x_i : A_i$ and it provides some additional information on the labels.

[2]The property of uniformity (in the sense of Miller's uniform proof paradigm) might be the base of a possible answer: goal-directed (uniform) proofs are, or correspond, to a very restricted kind of proof in a given sequent calculus. When we search for a a uniform proof we hence explore a restricted search space of possible derivations in the sequent calculus.

To give an example, a label may represent a position (or a state) and the database itself specifies which positions are accessible from any other. This information is expressed as a relational theory about a predicate $R(x, y)$ which can be interpreted as 'x sees y', or 'y is accessible from x'. This correspond to a modal reading of the data, the implication behaves as atrict implication in modal logic. Goals are proved relatively to a specific prosition, so that we will write a query as

$$\Delta \vdash^? x : G,$$

where x is a position. To keep the things simple, we consider here only Horn-clauses, and we put the following constraints:

- (success rule) $\Delta \vdash^? x : q$ succeeds if $x : q \in \Delta$;
- (reduction rule) From $\Delta \vdash^? x : q$ step to $\Delta \vdash^? x : G$ if there is $y : G \to q \in \Delta$ such that $R(y, x)$.

A similar modal reading of clauses is at the base of the modal-logic programming language elaborated by Giordano, Martelli and Rossi [Giordano et al. 92],[Giordano and Martelli 94]. Modal operators are used to govern visibility rules within logic programs. In this way, it is possible to introduce structuring concepts, such as modules and blocks in logic programs, basing on logic.

Example 1.2.2 Consider the following database Δ with the data

$$\begin{array}{l} x : b \to c, \\ x : d \to c, \\ y : a, \\ z : a \to b, \\ z : d \end{array}$$

Moreover, we know that $R(x, y), R(y, z)$ and that R is symmetric. Suppose, we want to check if Δ proves c at position y, that is $\Delta \vdash^? y : c$. This configuration can be displayed as in Figure 1.2

$$\begin{tikzcd}
x :\ b \to c, d \to c \arrow[d, dash] \\
y :\ a \vdash^? c \arrow[d, dash] \\
z :\ a \to b, d
\end{tikzcd}$$

Figure 1.2:

The query succeeds:

$$\begin{array}{l} \Delta \vdash^? y : c \\ \Delta \vdash^? y : b, \text{ since } x : b \to c \in \Delta \text{ and } R(x, y), \\ \Delta \vdash^? y : a, \text{ since } z : a \to b \in \Delta \text{ and } R(z, y), \text{ by simmetry from } R(y, z). \end{array}$$

The last query immediately succeeds, as $y : a \in \Delta$. Notice however that neither $\Delta \vdash^? x : c$, nor $\Delta \vdash^? z : c$ succeeds.

This example may seem arbitrary, but the restrictions on success and reduction have a logical meaning. If we interpret $\to$ as *modal strict implication* $\Rightarrow$, that is $(A \Rightarrow B) =_{def} \Box(A \to B)$, the above configuration is (essentially) what is built in a proof of the formula

$$[(b \Rightarrow c) \wedge (d \Rightarrow c) \wedge ((a \Rightarrow b) \wedge d) \Rightarrow e)] \Rightarrow (a \Rightarrow c)\ ^{3}.$$

This formula happens to be a theorem of modal logic K5. ■

Another interpretation of the labels is that they represent *resources* and the deduction process put some constraint about their *usage*. Example of constraints about usage are:

- we may require that we use all data (or a specified subset of the data).
- we may require that we use the data no more than once.
- we may require that we use the data in a given order.

These constraints correspond to well-known logics, namely the so-called substructural logics [Gabbay 96],[Anderson and B
In this case, a query will have the form

$$\Gamma \ \vdash^{?} \ \alpha : G,$$

where α is an ordered set of atomic labels representing resources. Confining ourself to the Horn-case, we can put, as an example, the following constraints

- (success rule) $\Delta \ \vdash^{?} \ \alpha : q$ succeeds if $\alpha = \{x\} \wedge x : q \in \Delta$;
- (reduction rule1) From $\Delta \ \vdash^{?} \ \alpha : q$ step to $\Delta \ \vdash^{?} \ \alpha - \{y\} : G$ if there is $y : G \to q \in \Delta$ and $y \in \alpha$.
- (and rule1) From $\Delta \ \vdash^{?} \ \alpha : A \wedge B$ step to $\Delta \ \vdash^{?} \ \alpha_1 : A$ and $\Delta \ \vdash^{?} \ \alpha_2 : B$, provided $\alpha_1 \cap \alpha_2 = \emptyset$ and $\alpha_1 \cup \alpha_2 = \alpha$.

Another example of constraints is the following, we denote by $\max(\alpha)$ the maximal label in α and we set:

- (reduction rule2) From $\Delta \ \vdash^{?} \ \alpha : q$ step to $\Delta \ \vdash^{?} \ \alpha' : G$ if there is $y : G \to q \in \Delta$ such that
  $$y \in \alpha \ \wedge \ y \leq max(\alpha') \wedge \alpha' \cup \{y\} = \alpha.$$
- (and rule2) From $\Delta \ \vdash^{?} \ \alpha : A \wedge B$ step to $\Delta \ \vdash^{?} \ \alpha_1 : A$ and $\Delta \ \vdash^{?} \ \alpha_2 : B$, provided $max(\alpha_1) \leq max(\alpha_2)$ and $\alpha_1 \cup \alpha_2 = \alpha$.

Example 1.2.3 Let us call P1 the proof system with restrictions 1 and P2 the proof systems with restrictions 2. Let Δ_1 be the following database:

$x_1 : d$
$x_2 : a \wedge b \to c$
$x_3 : d \to a$
$x_4 : b.$

Figure 1.3 shows a succeesful derivation of $\Delta_1 \ \vdash^{?} \ \{x_1, x_2, x_3, x_4\} : c$ according to procedure P1. We omit the database since it is fixed. This query fails under procedure P2, step (*) violates the constraint in the reduction rule2 rule.

On the other hand, let Δ_2 be the following database:

[3]The configuration above is simplified, because we do not want to handle implicational goals in this example. The actual configuration generated in the proof of this formula contains also $x : ((a \Rightarrow b) \wedge d) \Rightarrow e$.

$$\begin{tikzcd}
 & \vdash^{?} \{x_1, x_2, x_3, x_4\} : c \arrow[d, dash] & \\
 & \vdash^{?} \{x_1, x_3, x_4\} : a \wedge b \arrow[dl, dash] \arrow[dr, dash] & \\
(*) \ \vdash^{?} \{x_1, x_3\} : a \arrow[d, dash] & & \vdash^{?} \{x_4\} : b \\
\vdash^{?} \{x_1\} : d & &
\end{tikzcd}$$

Figure 1.3:

$x_1 : a \wedge b \to c$
$x_2 : d \to a$
$x_3 : d \to b$
$x_4 : d.$

Figure 1.4 shows a succeesful derivation of $\Delta_2 \ \vdash^{?} \ \{x_1, x_2, x_3, x_4\} : c$ according to procedure P2. We omit the database since it is fixed. This query fails under procedure P1, step (*) violates the constraint in the (and rule1) rule.

$$\begin{tikzcd}
 & \vdash^{?} \{x_1, x_2, x_3, x_4\} : c \arrow[d, dash] & \\
 & (*) \ \vdash^{?} \{x_2, x_3, x_4\} : a \wedge b \arrow[dl, dash] \arrow[dr, dash] & \\
\vdash^{?} \{x_2, x_4\} : a \arrow[d, dash] & & \vdash^{?} \{x_3, x_4\} : b \arrow[d, dash] \\
\vdash^{?} \{x_4\} : d & & \vdash^{?} \{x_4\} : d
\end{tikzcd}$$

Figure 1.4:

As in the case of modal logic, the restrctions we have put in the rules may seem arbitrary, but they correspond to well-known logic. If we interpret $\to$ as the linear implication $-\circ$ and $\wedge$ as the intensional conjucntion $\otimes$, procedure P1 is complete for a (Horn) $-\circ, \otimes$-fragment of linear logic, and the success of the former query shows the validity in linear logic of the formula

$$[d \otimes (a \otimes b - \circ c) \otimes (d - \circ a) \otimes b] - \circ c.$$

In a similar way, if we interpret $\to$ as relevant implication and $\wedge$ as the relevant conjunction $\circ$, procedure P2 is complete for a (Horn) $\to, \circ$-fragment of relevant logic T (Ticket Entailment), and the success of the latter query shows the validity in T of the formula

$$[(a \circ b \to c) \circ (d \to a) \circ (d \to b) \circ d] \to c.$$

$\blacksquare$

The methodology of controlling the deduction process by labelling data and then putting constraints on the propagation and the composition of labels is very powerful, it is extensively studied in [Gabbay 96].

We have started from Horn deduction, we have added intuitionistic implication and then we have turned to strict implication modal logic and substructural logics. We still do not know what is the place of classical logic in this framework. If we confine ourself to Horn clauses, the basic deduction procedure we have described is complete for both intutionistic and classical provability. If we allow nested implications, this is no longer true.

For instance, let us consider Peirce's law:

$(a \to b) \to a \ \vdash^? \ a$ reduce to
$(a \to b) \to a \ \vdash^? \ a \to b$, which reduce to
$(a \to b) \to a, a \ \vdash^? \ b$.

In intuitionistic logic we fail at this point because b does not unify with the head of any formula/clause. We know that in classical logic, this formula must succeed since it is a tautology. What can we do? The answer is simple, we carry on the computation by re-asking the original goal:

$$(a \to b) \to a, a \ \vdash^? \ a$$

We immediately succeed. To get classical logic, we just need a rule which allows to replace the current (atomic) goal by a previous one. To this purpose, the structure of queries must be enriched to record the history of past goals, a query will have the form

$$\Gamma \ \vdash^? \ G, H$$

where H is the sequence of past goals. In case of classical logic, we can limit to record atomic goals, and we record them when we perform a reduction step. With this book-keeping the previous derivation becomes:

(1) $(a \to b) \to a \ \vdash^? \ a, \emptyset$
(2) $(a \to b) \to a \ \vdash^? \ a \to b, \{a\}$
(3) $(a \to b) \to a, a \ \vdash^? \ b, \{a\}$
(4) $(a \to b) \to a, a \ \vdash^? \ a, \{a\}$ by restart

In case of classical logic this simple restart rule can be easily understood in terms of standard Gentezen calculi. It correspond to allowing both *weakening and contraction on the right*. A sequent derivation corresponding to the above one is as follows:

$$\cfrac{\cfrac{\cfrac{\cfrac{(4)\ a \vdash a}{(3)\ a \vdash a, b}}{(2)\ \vdash a \to b, a} \qquad a \vdash a}{(1')\ (a \to b) \to a \vdash a, a}}{(1)\ (a \to b) \to a \vdash a}$$

It can be seen that both weakening (to get (3) from (4)) and contraction on right (to get (1) from (1')) are required. In other words, the formula we restart from is connected by a disjunction to the current goal. The idea of restart has been first proposed by Gabbay in his lecture notes in 1984 [Gabbay 84] and the theoretical results published in [Gabbay 85]. The lecture notes have evolved into the book [Gabbay 98].

A similar idea to restart has been exploited by Loveland [Loveland 91],[Loveland 92], in order to extend conventional logic programming to non-Horn databases; In Loveland's proof-procedure the restart rule is a way of implementing reasoning by case-analysis.

In case of classical logic it is simple to give a translation of the restart rule in terms of the rules of a sequent claculus. In some other cases, restart rules may take into account not only previous goals, but also the relative databases (or the positions) from which they were asked. In these cases, there may be no counterpart of restart rules in terms of sequent rules. This for instance the case of Dummett's logic LC presented in chapter 3.

1.3 Goal-directed Analysis of Logics

Goal directed procedures are analytic and cut-free. One can often prove a cut admissibility property of computations and to establish further properties such as interpolation. The general form of cut is the following: if the two queries

$$\Gamma[A] \vdash^? B,$$

$$\Delta \vdash^? A$$

succeed in the goal directed procedure than also

$$\Gamma[A/\Delta] \vdash^? B$$

succeeds. Thus, the notion of cut depends on the notion of substitution. For each logic, the specific property of cut is determined by the conditions on the substitution operation $\Gamma[A/\Delta]$ which might be different for each logic, even if the structure of databases is the same [4]. To see an example, let us consider *the pure strict implicational* fragment of modal logics K and S4, for more details see chapter 4. In this case, the database can be assumed to be a list of formulas. given

$$\Gamma = C_1, \ldots, C_{i-1}, A, C_{i+1}, \ldots, C_n$$

we can replace A by Δ if

$$\Delta = C_1, \ldots, C_{i-1}, \mathbf{B}$$

and the result of the substitution is

$$\Gamma[A/\Delta] = C_1, \ldots, C_{i-1}, \mathbf{B}, C_{i+1}, \ldots, C_n.$$

Thus, if

$\Gamma = C_1, \ldots, C_{i-1}, \mathbf{A}, C_{i+1}, \ldots, C_n \vdash^? D$ and $\Delta = C_1, \ldots, C_{i-1}, \mathbf{B} \vdash^? \mathbf{A}$ both succeed

then we can cut and obtain that

$\Gamma = C_1, \ldots, C_{i-1}, \mathbf{B}, C_{i+1}, \ldots, C_n \vdash^? D$ succeeds too.

In case of S4, we are more liberal: substitution and hence cut is also allowed whenever

[4]For a discussion on cut and structural consequence relations we refer to [Gabbay 93] and [Avron 91b].

$\Delta = C_1, \ldots, C_{i-1}, \mathbf{B_1}, \ldots, \mathbf{B_t}$, with $t \geq 0$.

If $\Delta = C_1, \ldots, C_{i-1}, \mathbf{A}, C_{i+1}, \ldots, C_n \vdash^? D$, we get now that

$\Gamma = C_1, \ldots, C_{i-1}, \mathbf{B_1} \ldots, \mathbf{B_t}, C_{i+1}, \ldots, C_n \vdash^? D$ succeeds too.

Other examples of constrained cut occur int the case of substructural logics. In this case the goal and the database are labelled and we will have to impose specific conditions on the label of the cut-formula, the labels of the goal, and the database.

The reason why cut elimination process works well with the goal-directed style of deduction is that the deduction rules strongly constraint the form of derivations: there is no other way to prove a goal than decomposing it until we reach its atomic constituents which are then matched against database formulas. Moreover, unlike Gentzen systems, in goal-directed procedures there are no separate structural rules. These rules are, so to say, incorporated in the other computation rules.

The cut-admissibility property turns out to be the essential step needed to prove the completeness of the procedure. Namely, in most of the cases, the specific cut admissibility property for each logic is *equivalent* to the completeness of the procedure with respect to the canonical model for that logic as it is determined by the deduction procedure itself. This relation between cut-elimination and completeness has been pointed out by Miller in [Miller 92].

On the other hand, we can reverse the relation between semantics and proof procedure. Given a proof-procedure P, we can always define

$\Delta \vdash_P A \Leftrightarrow \Delta \vdash^? A$ succeeds in P.

Usually, the goal-directed procedure P satisfies some form of cut elimination and hence it defines a consequence relation $\vdash_P$ which is closed under cut, and may have other properties (such as some form of identity and monotonicity). Instead of asking if $\vdash_P$ is complete with respect to an already-known semantics, we may ask whether there is a closely-related "semantic counterpart" for the consequence relation $\vdash_P$ defined as above. We will see an example in chapter 4, where we have some particular deduction procedures for which cut is admissible define some intuitionistic modal logics.

1.4 Outline of the book

- In **chapter 2** we define goal-directed procedures for intuitionistic and classical logics. We start with the implicational fragment. Then, we study how to optimize the procedure by avoiding re-use of data. This is also needed to make it terminating. We start with the implicational fragment, and then we consider richer propositional fragments and finally the implication-universal quantifier fragment of intuitionistic logic.

- In **chapter 3**, we treat some implicational intermediate logics. We first consider the logics of Kripke models with bounded height. The we treat one of the most important intermediate logic: Gödel-Dummett's logic LC. This logic is complete with respect to linear Kripke models of intuitionistic logic and has also a natural many-valued semantics.

- In **chapter 4**, we define goal directed procedures for strict implication as determined by several modal logics. By strict implication, we intend the connective $\Rightarrow$ defined by $A \Rightarrow B = \Box(A \rightarrow B)$, where the modality is understood according to each specific modal system. Our proof systems covers uniformly strict implication of well-known modal logics K, T, K4, S4, K5, K45, and S5. We also explore some intutionistic variants, and we finally give a procedure for Gödel modal logic of provability G.

- In **chapter 5** we define goal directed procedures for the most important implicational substructural logics, namely R, Linear Logic, Lambek calculus, and other relevant logics such as Ticket Entailment and E. For logics without contraction, these procedures are also decision procedure, whereas for those logics which allow contraction they are not. We show how to turn the procedure for implicational R into a decision procedure by adding a loop-checking mechanism.

- In **chapter 6** we highlight possible directions of further research, listing a number of open problems which we are going to deal in future work.

1.5 Notation and basic notions

In this section we introduce some basic notations and notions we will use thoughout the book. Other notions will be introduced when needed in their proper place.

Formulas
By a propositional language $\mathcal{L}$, we denote the set of propositional formulas built out from a denumerable set Var of propositional variables by applying the propositional connectives $\neg, \wedge, \vee, \rightarrow$.

Unless stated otherwise, we denote propositional variables (also called *atoms*) by lower case letters, and arbitrary formulas by upper case letters.
We assign a *complexity* $cp(A)$ to each formula A (as usual):

$cp(q) = 0$ if q is an atom,
$cp(\neg A) = 1 + cp(A)$,
$cp(A * B) = cp(A) + cp(B) + 1$, where $* \in \{\wedge, \vee, \rightarrow\}$.

(*Formula substitution*) We define the notion of substitution of an atom q by a subformula B within a formula A. This operation is denoted by $A[q/B]$.

$$p[q/B] = \begin{cases} p \text{ if } p \neq q \\ B \text{ if } p = q \end{cases}$$

$$(\neg A)[q/B] = \neg A[q/B]$$

$$(A * C)[q/B] = A[q/B] \rightarrow C[q/B] \text{ where } * \in \{\wedge, \vee, \rightarrow\}$$

Implicational formulas
In great part of the work we will be concerned with *pure implicational formulas*. These formulas are generated from a set of atoms by the only connective $\rightarrow$. We adopt some specific notations for them. We sometimes distinguish the *head* and the *body* of an implicational formula. The head of a formula A is its rightmost nested atom, whereas the body is the *list* of the antecedents of its head. Given a formula A, we define $Head(A)$ and $Body(A)$ as follows:

$Head(q) = q$, if q is an atom,
$Head(A \rightarrow B) = Head(B)$.

$Body(q) = [\,]$, if q is an atom,
$Body(A \to B) = (A) * Body(B)$,

where $A * Body(B)$ denotes the list beginning with A followed by $Body(B)$.
Dealing with implicational formulas, we assume that implication associates on the right, i.e. we write

$A_1 \to A_2 \to \ldots \to A_{n-1} \to A_n$,

instead of $A_1 \to (A_2 \to \ldots \to (A_{n-1} \to A_n) \ldots)$.
It turns out that every formula A can be written as

$$A_1 \to A_2 \to \ldots \to A_n \to q,$$

where we obviously have.

$Head(A) = q \quad \text{and} \quad Body(A) = (A_1, \ldots, A_n)$

Multisets
A (finite) multiset is a function α from a (finite) set S to $\mathcal{N}$, the set of natural numbers. We denote multisets by greek letters $\alpha, \beta \ldots$. We define the following operations:

(Union) $\alpha \sqcup \beta = \gamma$ iff $\forall x \in S\ \gamma(x) = \alpha(x) + \beta(x)$.

(Difference) $\alpha - \beta = \gamma$ iff $\forall x \in S\ \gamma(x) = \alpha(x) - \beta(x)$, where "$-$" is subtraction on natural numbers.

The *support* of a multiset α, denoted by $\bar{\alpha}$ is the set of $x \in S$, such that $\alpha(x) > 0$.
In order to display the elements of a multiset α, we use the notation

$\alpha = [x, x, y, z, z, z]$, or equivalently $[x^2, y, z^3]$,

which means that $\alpha(x) = 2, \alpha(y) = 1, \alpha(z) = 3$, and $\bar{\alpha} = \{x, y, z\}$.
(*Weak containment*) We define $\alpha \subseteq \beta$ as follows

$\alpha \subseteq \beta$ to hold iff $\forall x \in S\ \alpha(x) \leq \beta(x)$.

(*Strong containment*) We define the relation $\alpha \subseteq\!| \beta$ as follows

$\alpha \subseteq\!| \beta$ iff $\alpha \subseteq \beta$ and $\bar{\alpha} = \bar{\beta}$.

We use the notation $\alpha \subset \beta$ and $\alpha \subset\!| \beta$, for the corresponding strict versions of the relations defined above.

Chapter 2

Intuitionistic and Classical logics

COMMENT: WE MUST ADD SOMETHING HERE TO MOTIVATE THE IMPORTANCE OF INTUITIONISTIC LOGIC

In this chapter we present a goal-directed proof system for intuitionistic logic. The chapter proceeds to discover the algorithmic system. We start from the implicational fragment and we stepwise refine and extend it. Each new step will follow naturally from the previous steps. We further modify it and get classical logic. In the next section we present intuitionistic logic giving an axiom system and a model-theory.

[ADD STANDARD REFERENCES ON INTUITIONISTIC LOGIC]

2.1 Alternative presentations of Intuitionistic Logic

There are many ways of presenting intuitionistic logic. We begin giving an Hilbert style axiomatization of the propositional calculus using the following set of axioms, we denote by **I**:

(1) Implication group:

1. $A \to A$,
2. $(A \to B) \to (C \to A) \to C \to B$,
3. $(A \to B) \to (B \to C) \to A \to C$,
4. $(A \to B \to C) \to B \to A \to C$,
5. $(A \to A \to B) \to A \to B$,
6. $(A \to B \to C) \to (A \to B) \to A \to C$,
7. $A \to B \to A$.

(2) Conjunction group:

1. $A \to B \to (A \wedge B)$,
2. $A \wedge B \to A$,
3. $A \wedge B \to B$.

(3) Disjunction group:

1. $(A \to C) \to (B \to C) \to (A \vee B \to C)$,
2. $A \to A \vee B$,
3. $A \to B \vee A$.

(4) Falsity:

1. $\bot \to A$,

(5) Negation group:

1. $\neg A \to A \to \bot$,
2. $(A \to \neg B) \to B \to \neg A$,
3. $\neg A \to A \to B$.

It contains in addition Modus Ponens Rule:

$$\frac{\vdash A \quad \vdash A \to B}{\vdash B},$$

This axiom system is *separated*, that is to say, any theorem containing $\to$ and a set of connectives $S \subseteq \{\wedge, \vee, \neg, \bot\}$ can be proved by using the implicational axioms together with the axiom groups containing just the connectives in S.

Furthermore, the implicational axioms are not independent. For instance, the axioms $A \to B \to A$ and $(A \to B \to C) \to (A \to B) \to A \to C$ are sufficient to prove the remaining implicational axioms. However without anyone of the two, we can get various weaker systems (some of them known as substructural logics, see chapter 5) by dropping some of the other axioms. Another redundancy are negation and falsity axioms, as this two logical constants are interdefinable. One can adopt the axiom for falsity and define $\neg A$ as $A \to \bot$. Or the other way around, one can adopt the axioms for negation and consider $\bot$ as defined by, for instance $\neg(p_0 \to p_0)$, or $p_0 \wedge \neg p_0$, where p_0 is any atom. If we adopt *both* the axioms for negation and for $\bot$, we can *prove* their interdefinability, namely

$$\neg A \leftrightarrow (A \to \bot)$$

is a theorem of the above axiom system. We leave to check it to the reader.

HOWEVER THE PROOF OF THE HALF $(A \to \bot) \to \neg A$ IS NOT TRIVIAL !! SHALL WE PUT IT?

If add to **I**any of the axioms below we get classical logic:

(Peirce's law) $((A \to B) \to A) \to A$
(double negation) $\neg\neg A \to A$,
(excluded middle) $\neg A \vee A$,
(what name???) $[(A \to (B \vee C)] \to [(A \to B) \vee C]$.

In particular, the addition of Peirce's law to the implicational axioms of intuitionistic logic give us an axiomatization of classical implication.

We introduce a standard model-theoretic semantics of intuitionistic logic, called Kripke semantics:

Definition 2.1.1 A Kripke model is a structure of the form $M = (S, \leq, a, V)$, where S is a non-empty set, $\leq$ is a reflexive and transitive relation on S, $a \in S$, $V : S \to Pow(Var)$, that is maps each element of S to a set of propositional variables. We assume the following conditions:

(1) $a \leq x$, for all $x \in S$;

(2) $x \leq y$ implies that $V(x) \subseteq V(y)$;

(3) $\perp \notin V(x)$, for all $x \in V$.

Truth condition are given through the following clauses:

- $M, x \models q$ iff $q \in V(x)$l;
- $M, x \models A \wedge B$ iff $M, x \models A$ and $M, x \models B$;
- $M, x \models A \vee B$ iff $M, x \models A$ or $M, x \models B$;
- $M, x \models A \to B$ iff for all $y \geq x$, if $M, y \models A$ then $M, y \models B$;
- $M, x \models \neg A$ iff for all $y \geq x$, if $M, y \not\models A$.

We say that A is true in M if $M, a \models A$ and we denot it by $M \models A$. We say that A is valid if it is true in every Kripke model M. We also define a notion of entailment between sets of formulas and formulas. Let $\Delta = \{A_1, \ldots, A_n\}$ be a set of formulas and B be a formula, we say that Γ *entails* B denoted by $\Gamma \models B$ iff

$M, a \models A_i$ for all $A_i \in \Gamma$, then $M, a \models B$.

■

One can think of a Kripke model M as above as a tree with root a. The definition of entailment is by no means restricted to finite Γ's. Whenever Γ is finite, $\Gamma \models A$ holds iff $\bigwedge \Gamma \to A$ is valid. It is easy to prove that condition (2) implies that in every model M, for any x, y, if $x \leq y$ and $M, x \models A$, then $M, y \models A$. By this property, we immediately obtain:

$M, a \models A$ iff for all $x \in S$ $M, x \models A$

By this fact we can equally define truth in a model as truth in *every* point of the model.

A standard argument shows that the propositional calculus given above is sound and complete with respect to Kripke models.

Theorem 2.1.2 (REFERENCES???) *for any formula A, A is a theorem of I iff is valid in every Kripke model.*

This completeness theorem can be sharpened to *finite* Kripke models (finite trees), that is models $M = (S, \leq, a, V)$, where S is a finite set.

Theorem 2.1.3 (REFERENCES???) *for any formula A, A is a theorem of I iff is valid in every* finite *Kripke model.*

Classical interpretations can be thought as degenerated Kripke models $M = (S, \leq, a, V)$, where $S = \{a\}$.

We give a third presentation of intuitionistic logic in terms of consequence relation. Let Γ denote a set of formulas. We write

$$\Gamma \vdash A$$

to say that Γ proves A, where A is a formula. We will often use "," to denote set-theoretic union, i.e. we write Γ, A and Γ, Δ to denote $\Gamma \cup \{A\}$ and $\Gamma \cup \Delta$.

Definition 2.1.4 [Consequence relation for intuitionistic logic] Let $\Gamma \vdash A$ be defined as the smallest relation which satisfy:

- (identity) if $A \in \Gamma$ then $\Gamma \vdash A$;
- (monotony) if $\Gamma \vdash A$ and $\Gamma \subseteq \Delta$ then $\Delta \vdash A$;
- (cut) $\Gamma \vdash A$ and $\Gamma, A \vdash B$ imply $\Gamma \vdash B$;

Plus the following conditions for the language containing $\{\wedge, \vee, \rightarrow, \bot\}$

1. Deduction theorem $\Delta, A \vdash B$ iff $\Delta \vdash A \rightarrow B$
2. Conjunction rules
   (a) $A \wedge B \vdash A$
   (b) $A \wedge B \vdash B$
   (c) $A, B \vdash A \wedge B$.
3. Falsity Rule $\bot \vdash B$.
4. Disjunction rules
   (a) $A \vdash A \vee B$
   (b) $B \vdash A \vee B$
   (c) $\Delta, A \vdash C$ and $\Delta, B \vdash C$ imply $\Delta, A \vee B \vdash C$.

■

The above closure rules define the intuitionistic propositional consequence relation. In the fragment without $\bot$, the other rules (1), (2), and (4) suffice to define intuitionistic logic for that fragment. In the above characterization, we negation is not mentioned. We can either consider negation as defined by $\neg A = A \rightarrow \bot$, or add the rules:

$$A, \neg A \vdash B$$

$$\frac{\Gamma, A \neg B}{\Gamma, B \vdash \neg A}$$

Classical logic can be obtained as the smallest consequence relation satisfying (1)–(4) and an additional condition such as (5) below:

6. Strong deduction theorem
$$\Delta, A \vdash B \lor C \text{ iff } \Delta \vdash (A \to B) \lor C.$$

IT IS CLEAR THAT THERE ARE OTHER WAYS OF OBTAINING CLASSICAL CONSEQUENCE RELATION, FOR INSTANCE BY ADDING:

$\Gamma, A \vdash B$ and $\Gamma, \neg A \vdash B$ implies $\Gamma \vdash B$

SHALL WE MENTION IT???

Theorem 2.1.5 (REFERENCES???) *Let $\vdash$ be the smallest consequence relation satisfying (1) - (4) above. Then $\Gamma \vdash A$ holds iff $\Gamma \models A$.*

2.2 Rules for intuitionistic implication

We want to give computation rules for checking $\Delta \vdash A$, where all formulas of Γ and A are implicational. Our rules manipulates queries Q of the form:

$\Delta \ \vdash^? \ A$,

We call Δ the *database* and A the goal of the query Q. We use the symbol $\vdash^?$ to indicate that we do not know whether the query succeeds or not, on the other hand the success of Q means that $\Delta \vdash A$ according to intuitionistic logic. Of course this must be proved and it will in the due course, in the meanwhile here are the rules.

Definition 2.2.1
- (success) $\Delta \ \vdash^? \ q$ succeeds if $q \in \Delta$. We say that q is used in this query.
- (implication) from $\Delta \ \vdash^? \ A \to B$ step to

  $\Delta, A \ \vdash^? \ B$
- (reduction) from $\Delta \ \vdash^? \ q$
  if $C \in \Delta$, with $C = D_1 \to D_2 \to \ldots \to D_n \to q$
  (that is $Head(C) = q$ and $Body(C) = \{D_1, \ldots D_n\}$) then step to

  $\Delta \ \vdash^? \ D_i$, for $i = 1, \ldots, n$.

  We say that C is used in this step.

■

A *derivation* $\mathcal{D}$ of query Q is a tree whose nodes are queries. The root of $\mathcal{D}$ is query Q, and the successors of every non-leaf query are determined by exactly one applicable rule (*implication or reduction*) as described above.
We say that $\mathcal{D}$ is *successful* if *success rule* may be applied to every leaf of $\mathcal{D}$.
We finally say that a query Q *succeeds* if there is a successful derivation of Q.

By definition, a derivation $\mathcal{D}$ might be an infinite tree, however if $\mathcal{D}$ is successful , then must be finite. This is easily seen from the fact that, in case of success, the height of $\mathcal{D}$ is finite and every non-terminal node of $\mathcal{D}$ has a finite number of successors, because of the form of the rules. Moreover, the databases involved in a deduction need not be finite. In a successful derivation only a finite number of formulas from the database will be used in the sense mentioned above.

A last observation: the success of a query is defined in a non-deterministic way; a query succeeds if there is a successful derivation. To transform the proof rules in a deterministic algorithm one should give a method to search a successful derivation tree. In this respect we agree that when we come to an atomic goal we try to apply first the success rule and if it fails we try the reduction rule. Then the only choice is which formula of the database whose head matches the current atomic goal we use to perform a reduction step, if there are more than one. Thinking the database as a list of formulas, we can choose the first one and remember the point up to which we have scanned the database as a backtracking point. This is exactly as in conventional logic programming [Lloyd 84].

Example 2.2.2 We check that

$$b \to d, a \to p, p \to b, (a \to b) \to c \to a, (p \to d) \to c \vdash b$$

Let $\Gamma = \{b \to d, a \to p, p \to b, (a \to b) \to c \to a, (p \to d) \to c\}$, a successful derivation of $\Gamma \vdash^? b$ is shown in Figure 2.1. A quick explanation: (2) is obtained by reduction wrt. $p \to b$, (3) by reduction wrt. $a \to p$, (4) and (8) by reduction wrt. $(a \to b) \to c \to a$, (6) by reduction wrt. $p \to b$, (7) by reduction wrt. $a \to p$, (9) by reduction wrt. $(p \to d) \to c$, (11) by reduction wrt. $b \to d$, (12) by reduction wrt. $p \to b$.

$$\begin{tikzcd}
 & (1)\ \Gamma \vdash^? b \arrow[d, dash] & \\
 & (2)\ \Gamma \vdash^? p \arrow[d, dash] & \\
 & (3)\ \Gamma \vdash^? a \arrow[dl, dash] \arrow[dr, dash] & \\
(4)\ \Gamma \vdash^? a \to b \arrow[d, dash] & & (8)\ \Gamma \vdash^? c \arrow[d, dash] \\
(5)\ \Gamma, a \vdash^? b \arrow[d, dash] & & (9)\ \Gamma \vdash^? p \to d \arrow[d, dash] \\
(6)\ \Gamma, a \vdash^? p \arrow[d, dash] & & (10)\ \Gamma, p \vdash^? d \arrow[d, dash] \\
(7)\ \Gamma, a \vdash^? a & & (11)\ \Gamma, p \vdash^? b \arrow[d, dash] \\
 & & (12)\ \Gamma, p \vdash^? p
\end{tikzcd}$$

Figure 2.1:

■

We state some simple properties of the deduction procedure defined above.

Proposition 2.2.3 *(a)* $\Delta \vdash^? G$ *succeeds if* $G \in \Delta$*;*

(b) $\Delta \vdash^? G$ *implies that* $\Delta, \Gamma \vdash^? G$ *succeeds;*

(c) $\Delta \vdash^? A \to B$ *succeeds iff* $\Delta, A \vdash^? B$ *succeeds.*

Proof.

(a) is proved by induction on the complexity of G. If G is an atom q, it follows immediately by the success rule. If $G = A_1 \to \ldots \to A_n \to q$, then from $\Delta \vdash^? G$, we step, by repeated application of the implication rule to

$$\Delta, A_1, \ldots, A_n \vdash^? q.$$

Since $G \in \Delta$, we can apply reduction and step to

$$\Delta, A_1, \ldots, A_n \vdash^? A_i, \text{ for } i = 1, \ldots, n.$$

By induction hypothesis, each of the above queries succeeds.

(b) is proved by induction on the height of a successful computation. If G succeeds by the success rule, then G is atomic and $G \in \Delta$; hence $G \in \Delta, \Gamma$, thus $\Delta, \Gamma \vdash^? G$ succeeds. If G is an implication $A \to B$, then from $\Delta \vdash^? A \to B$, we step to $\Delta, A \vdash^? B$. In the same way from $\Delta, \Gamma \vdash^? A \to B$, we step to $\Delta, \Gamma, A \vdash^? B$, which succeeds by induction hypothesis. Let G be an atom q, suppose we proceed by reduction with respect to a formula, say, $C = A_1 \to \ldots \to A_n \to q$ in Δ, so that we step to

$$\Delta \vdash^? A_i, \text{ for } i = 1, \ldots, n,$$

then from $\Delta, \Gamma \vdash^? q$, we can perform the same reduction step with respect to C, and step to $\Delta, \Gamma \vdash^? A_i$, which succeed by induction hypothesis.

(c) is obvious, as there are no other rules, but the implication rule, which can be applied to an implicational goal.

□

We have called property (b) *monotony* of . It is clear from the proof that the height of derivations is preserved, that is if $\Gamma \vdash^? G$ succeeds by a derivation of height h, then also $\Delta, \Gamma \vdash^? G$ succeeds by a derivation of height h.

2.2.1 Soundness and Completeness

Since we are dealing with the pure implicational fragment, by theorem 2.1.3, it is enough to show that the consequence relation defined by:

$$\Delta \vdash_p A \ \leftrightarrow \ \Delta \vdash^? A \text{ succeeds}$$

coincides with the intuitionistic consequence relation. Let $\vdash$ denotes intuitionistic provability.

Theorem 2.2.4 *If* $\Delta \vdash_p A$ *then* $\Delta \vdash A$.

Proof. By induction on the height h of a successful derivation of $\Delta \vdash^? A$. Let $h = 0$. If $\Delta \vdash^? A$ succeeds by a derivation of height 0, then A is an atom and $A \in \Gamma$, thus $\Gamma \vdash A$ follows by reflexivity. Let $h > 0$, if $A = B \to C$, then $\Delta \vdash^? A$ succeed by the implication rule and $\Gamma, B \vdash^? C$ succeeds by a derivation of height $< h$, hence by induction hypothesis, $\Gamma, B \vdash C$ holds, and by the deduction theorem also $\Gamma \vdash B \to C$ holds. If A is an atom q, then the derivation proceeds by reduction with respect to a formula $C = A_1 \to \ldots \to A_n \to q$; then for $i = 1, \ldots, n$, each query

$\Gamma \vdash^? A_i$ succeeds by a derivation of height $< h$.

By induction hypothesis, for $i = 1, \dots, n$,

(a_i) $\Gamma \vdash A_i$ holds.

On the other hand, since $C \in \Gamma$, by reflexivity we have that

$\Gamma \vdash A_1 \to \dots \to A_n \to q$ holds,

so that, by deduction theorem also

(b) $\Gamma, A_1, \dots, A_n \vdash q$ holds.

By repeatedly applying cut to (a_i) and (b), we finally obtain that $\Gamma \vdash q$ holds. $\square$

In order to prove completeness it is sufficient to show that $\vdash_p$ satisfies the conditions of intuitionistic consequence relationship, namely identity, monotonicity, deduction theorem and cut. That the first three properties are satisfied is proved in proposition 2.2.3. So, the only property we have to check is cut. We prove closure under cut in the next theorem.

Theorem 2.2.5 (Admissibility of Cut) *If*

(1) $\Delta, A \vdash_p B$ *and (2)* $\Gamma \vdash_p A$ *then also*
$\Delta, \Gamma \vdash_p B$.

Proof. The theorem is proved induction on lexicographically- ordered pairs (c, h), where $c = cp(A)$, and h is the height of a successful derivation of (1), that is $\Delta, A \vdash^? B$. Suppose first $c = 0$, then A is an atom p, and we proceed by induction on h. If $h = 0$, B is an atom q and either $q \in \Delta$ or $q = p = A$. In the first case, the claim trivially follows by proposition 2.2.3. In the second case it follows by hypothesis (2) and proposition 2.2.3.

Let now $h > 0$, then (1) succeeds either by implication rule or by reduction. In the first case, we have that $B = C \to D$ and from $\Delta, A \vdash^? C \to D$ we step to $\Delta, A, C \vdash^? D$, which succeeds by a derivation h' shorter than h. Since $(0, h') < (0, h)$, by induction hypothesis we get that $\Delta, \Gamma, C \vdash^? D$, succeeds, whence $\Delta, \Gamma \vdash^? C \to D$ succeeds too. Let (1) succeeds by reduction with respect to a clause $C \in \Delta$. Since A is an atom, $C \neq A$. Then $B = q$ is an atom. We let $C = D_1 \to \dots \to D_k \to q$. We have that for $i = 1, \dots, k$

$\Delta, A \vdash^? D_i$ succeeds by a derivation of height $h_i < h$.

Since $(0, h_i) < (0, h)$, we may apply the induction hypothesis and obtain that

(a_i) $\Delta, \Gamma \vdash^? D_i$ succeeds, for $i = 1, \dots, k$.

Since $C \in \Delta \cup \Gamma$, from $\Delta, \Gamma \vdash^? q$ we can step to (a_i) and succeed. This concludes the case of $(0, h)$.

If c is arbitrary and $h = 0$ the claim is trivial. Let $c > 0$ and $h > 0$. The only difference with the previous cases is when (1) succeeds by reduction with respect to A. Let us see that case. Let

$A = D_1 \to \dots \to D_k \to q$ and $B = q$.

Then we have that for $i = 1, \dots, k$ $\Delta, A \vdash^? D_i$ succeeds by a derivation of height $h_i < h$. Since $(c, h_i) < (c, h)$, we may apply the induction hypothesis and obtain that

(b_i) $\Delta, \Gamma \vdash^? D_i$ succeeds for $i = 1, \ldots, k$.

By hypothesis (2) we can conclude that

(3) $\Gamma, D_1, \ldots, D_k \vdash^? q$ succeeds by a derivation of arbitrary height h'.

Notice that each D_i has a smaller complexity than A, that is $cp(D_i) = c_i < c$. Thus $(c_1, h') < (c, h)$, and we can cut on (3) and (b_1), so that we obtain that

(4) $\Delta, \Gamma, D_2, \ldots, D_k \vdash^? q$ succeeds with some height h''.

Again $(c_2, h'') < (c, h)$, so that we can cut (b_2) and (4). By repeating the same argument up to k we finally obtain that

$\Delta, \Gamma \vdash^? q$ succeeds.

This concludes the proof. □

We can now easily prove completeness of our procedure with respect to intuitionistic consequence relation.

Theorem 2.2.6 *If* $\Delta \vdash A$, *then* $\Delta \vdash_p A$.

Proof. We know that $\vdash$ is the smallest consequence relation satisfying (identity), (monotonicity), (deduction theorem) and (cut). By proposition 2.2.3 and theorem 2.2.5, $\vdash_p$ satisfies these properties as well. Then, the claim follows by the minimality of $\vdash$. □

2.2.2 Interpolation

The meaning of the interpolation property is that whenever we have $\Gamma \vdash A$, we can find an intermediate formula B which does not contain any "concept" that is not involved either in Γ or in A, such that both

$$\Gamma \vdash B \quad \text{and} \quad B \vdash A.$$

B is called an interpolant of A and Γ. The interpolation property is a strong analytic property of deductions and it says that we can simplify a proof of $\Gamma \vdash A$ in intermediate steps which involve only non-logical constants (here propositional variables) which are present both in A and in Γ. In the pure implicational fragment, the interpolant cannot be assumed to be *a single* formula, consider for example

$a, b \vdash (a \to b \to q) \to q$,

there is no pure implicational formula A containing only the atoms a and b such that

$a, b \vdash A$ and $A \vdash (a \to b \to q) \to q$.

We must allow the interpolant to be a database itself. The right notion is hence the one of *interpolant database*, and the property of interpolation must be restated as follows: if $\Gamma \vdash A$, then there is a database Π, which contains only propositional symbols common to A and Γ, such that

$$\Gamma \vdash^* \Pi \quad \text{and} \quad \Pi \vdash A.$$

The relation $\vdash^*$ denotes provability among databases; in this case it can be simply defined as

$$\Gamma \vdash^* \Delta \ \Leftrightarrow\ \forall C \in \Delta \quad \Gamma \vdash C.$$

Obviously $\Gamma \vdash^* \{A\}$ is the same as $\Gamma \vdash A$.

Before we prove this property, we warn the reader that the stated property is not true in general. It may happen, that $\Gamma \vdash A$ but A and Γ do not share any propositional symbol, it is the case for instance of

$$p \vdash q \rightarrow q$$

where $p \neq q$. In this case, A (here $q \rightarrow q$) is a theorem by itself. It is clear that here Π should be the empty set of formulas, or the empty conjunction. We add a symbol to the language $\top$ to denote truth. We assume that $\top$ is in the language of any formulas. Actually $\top$ may be defined as $p_0 \rightarrow p_0$, where p_0 is a fixed atom; we have that

$\Gamma \vdash \top$, and hence
$\Gamma, \top \vdash A$ iff $\Gamma \vdash A$.

Thus the constant $\top$ represents the empty database in the object language. We can regard $\top$ as an atom which immediately succeeds from any database.

We can now define the language of a formula and of a database as follows: we let

$\mathcal{L}(A) = \{p : \ p \text{ is an atom and } p \text{ occurs in } A\} \cup \{\top\}$
$\mathcal{L}(\Delta) = \bigcup_{A \in \Delta} \mathcal{L}(A)$.

We also write $\mathcal{L}(A, B)$ for $\mathcal{L}(\{A, B\})$, and $\mathcal{L}(\Delta, A)$ for $\mathcal{L}(\Delta \cup \{A\})$.

The interpolation property is a simple consequence of the following lemma.

Lemma 2.2.7 *Let Γ and Δ be databases. If $\Gamma, \Delta \vdash q$ there is a database Π such that the following hold:*

1. $\Gamma \vdash^* \Pi$,
2. $\Pi, \Delta \vdash q$,
3. $\mathcal{L}(\Pi) \subseteq \mathcal{L}(\Gamma) \bigcap \mathcal{L}(\Delta, q)$.

Proof. We proceed by induction on the height h of a successful derivation of (1) $\quad \Gamma, \Delta \ \vdash^? \ q$.

Let $h = 0$, then either $q = \top$ or $q \in \Gamma \cup \Delta$. If $q = \top$ or $q \in \Delta - \Gamma$, then we take $\Pi = \{\top\}$, otherwise we take $\Pi = \{q\}$.

Let $h > 0$, then any successful derivation proceed by reduction of q either with respect to a formula of Δ (case a), or with respect to a formula of Γ (case b).

(Case a) let $C = D_1 \rightarrow \ldots \rightarrow D_u \rightarrow q \ \in \Delta$. The computation steps, for $i = 1, \ldots, u$, to $\Gamma, \Delta \ \vdash^? \ D_i$. Let us assume that $D_i = \ B_1^i \rightarrow \ldots \rightarrow B_{k_i}^i \rightarrow r_i$, (it might be $k_i = 0$, that is $D_i = r_i$), then, by the implication rule, the computation steps to

$\Gamma, \Delta, \{B_1^i, \ldots, B_{k_i}^i\} \ \vdash^? \ r_i$.

All the above queries succeed by a shorter derivation, thus by induction hypothesis, for each i there are databases Π_i such that

1. $\Gamma \vdash^* \Pi_i$,
2. $\Pi_i, \Delta, \{B_1^i, \ldots, B_{k_i}^i\} \vdash r_i$,
3. $\mathcal{L}(\Pi_i) \subseteq \mathcal{L}(\Gamma) \bigcap \mathcal{L}(\Delta, B_1^i, \ldots, B_{k_i}^i, r_i,)$.

Since D_i is part of $C \in \Delta$, we have that $\mathcal{L}(\Pi_i) \subseteq \mathcal{L}(\Gamma) \cap \mathcal{L}(\Delta)$. Let $\Pi = \bigcup_i \Pi_i$, then we have that $\Gamma \vdash^* \Pi$ and for $i = 1, \ldots, u$

$(*)$ $\Pi, \Delta, \{B_1^i, \ldots, B_{k_i}^i\} \vdash^? r_i$ succeeds

and also that $\mathcal{L}(\Pi_i) \subseteq \mathcal{L}(\Gamma) \bigcap \mathcal{L}(\Delta)$. $(*)$ implies that for each i $\Pi, \Delta \vdash^? D_i$ succeeds; since $C \in \Delta$ we can apply the reduction rule to q so that $\Pi, \Delta \vdash^? q$ succeeds. This concludes (case a).

(Case b) let $C = D_1 \to \ldots \to D_u \to q \in \Gamma$. The computation step for $i = 1, \ldots, u$ to $\Gamma, \Delta \vdash^? D_i$, and then to $\Gamma, \Delta, \{B_1^i, \ldots, B_{k_i}^i\} \vdash^? r_i$, where we assume $D_i = B_1^i \to \ldots \to B_{k_i}^i \to r_i$, (it might be $k_i = 0$, that is D_i might be an atom). All the above queries succeed by a shorter derivation, thus by induction hypothesis, for each i there are databases Π_i such that

1. $\Delta \vdash^* \Pi_i$,
2. $\Pi_i, \Gamma, \{B_1^i, \ldots, B_{k_i}^i\} \vdash r_i$,
3. $\mathcal{L}(\Pi_i) \subseteq \mathcal{L}(\Delta) \bigcap \mathcal{L}(\Gamma, B_1^i, \ldots, B_{k_i}^i, r_i,)$.

Since D_i is part of $C \in \Gamma$, the last fact implies that $\mathcal{L}(\Pi_i) \subseteq \mathcal{L}(\Delta) \bigcap \mathcal{L}(\Gamma)$. As in (case a) we let

$\Pi = \bigcup_i \Pi_i = \{E_1, \ldots, E_n\}$,

and we get that

(4) $\Delta \vdash^* \Pi$, and for all $i = 1, \ldots, u$
(5) $\Pi, \Gamma, \{B_1^i, \ldots, B_{k_i}^i\} \vdash r_i$, and also that
(6) $\mathcal{L}(\Pi) \subseteq \mathcal{L}(\Delta) \bigcap \mathcal{L}(\Gamma)$.

We now let $G = E_1 \to \ldots \to E_n \to q$. From (4) we obtain that $G, \Delta \vdash q$, and from (5), we get that

(7) $\Pi, \Gamma \vdash D_i$.

Since $C \in \Gamma$, we can conclude that $\Pi, \Gamma \vdash q$, and hence $\Gamma \vdash G$. Finally from (6), since q occurs in Γ, we have that $\mathcal{L}(G) \subseteq \mathcal{L}(\Gamma) \cap \mathcal{L}(\Delta, q)$. This concludes the proof. □

Theorem 2.2.8 (Interpolation) *If $\Gamma \vdash A$, then there is a database Π such that*

$\Gamma \vdash^* \Pi$ *and* $\Pi \vdash A$, *and*
$\mathcal{L}(\Pi) \subseteq \mathcal{L}(\Gamma) \cap \mathcal{L}(A)$.

Proof. We proceed by case according to the form of A. If A is an atom, then we take $\Pi = A$. If A is a complex formula, let

$A = F_1 \to \ldots F_u \to p$.

By hypothesis, we have that $\Gamma, \{F_1, \ldots F_u\} \vdash^? p$ succeeds. By the previous lemma there is a database Π such that

(1) $\Gamma \vdash^* \Pi$
(2) $\Pi, \{F_1, \ldots F_u\} \vdash p$, and
(3) $\mathcal{L}(\Pi) \subseteq \mathcal{L}(\Gamma) \cap \mathcal{L}(F_1, \ldots, F_u, p)$.

From (2) we immediately conclude that $\Pi \vdash A$ and from (3) that $\mathcal{L}(\Pi) \subseteq \mathcal{L}(\Gamma) \cap \mathcal{L}(A)$. □

2.3 Bounded resource deduction for Implicational Logic

In the previous section, we have introduced a goal-directed proof method for the intuitionistic logic with pure implication. We want now to refine it further towards an automated implementation. We will be guided by key examples. The examples will show us difficulties to overcome in order to achieve an implementable system. We will respond to the difficulties by proposing to do the most obvious optimizations at hand. Slowly the computation steps will evolve and we will most naturally be led to consider new logics, which correspond to various optimizations. We will be happy to note that the new logics thus obtained are really logics we already know, and we thus realize that we have stumbled on proof systems for other well-known logics.

Consider the data and query below:

$$q \to q \ \vdash^? \ q$$

Clearly the algorithm has to know it is looping. The simplest way of loop checking is to record the history of the computation. This can be done as follows: Let $\Delta \ \vdash^? \ G, H$ represent the query of the goal G from data Δ and history H. H is the list of past queries, namely pairs of the form (Γ, G'). The computation rules become:

Historical rule for $\to$

$\Delta \ \vdash^? \ A \to B, H$ succeeds
if $\Delta \cup \{A\} \ \vdash^? \ B, H * (\Delta, A \to B)$ succeeds and $(\Delta, A \to B)$ is not in H.

Thus $(\Delta, A \to B)$ is appended to H.

Historical Reduction

$\Delta \ \vdash^? \ q, H$ succeeds,
if for some $B = C_1 \to C_2 \to \ldots C_n \to q$ in Δ we have that for all i
$\Delta \ \vdash^? \ C_i, H * (\Delta, q)$ succeeds and that (Δ, q) is not in H.

Thus the computation for our example above becomes:

$q \to q \ \vdash^? \ q, \emptyset$
$q \to q \ \vdash^? \ q, (q \to q, q)$
fail.

Note that the historical loop checking conjunct in *the rule for* $\to$ is redundant as the loop will be captured in *the rule for reduction*. Also note that in this case we made the decision that looping means failure. The most general case of loop checking may first detect the loop and then decide that under certain conditions looping implies success.

One way to perform loop-checking in a more efficient way has been described by Seyfried and Huerding in [Seyfried and Huerding 97]. The idea is to avoid repeated insertions of the same formula in the database, taking advantage of the fact that the database is a *set* of formulas. One simply records the history of the atomic goals asked from the same database; the history is cleared when a new formula is inserted in the database. This idea is implemented in a terminating proof system for full intuitionistic propositional logic in the form of a sequent calculus. We reformulate here this variant of loop checking adapted to our context.

An improved Historical loop-checking

Here H is the list of past atomic goals. The computation rules become:

Rule 1 for $\to$

$\Delta \vdash^? A \to B, H$ succeeds
if $A \notin \Delta$ and $\Delta, A \vdash^? B, \emptyset$ succeeds.

Rule 2 for $\to$

$\Delta \vdash^? A \to B, H$ succeeds
if $A \in \Delta$ and $\Delta \vdash^? B, H$ succeeds.

Reduction Rule

$\Delta \vdash^? q, H$ succeeds,
if $q \notin H$ and for some $C_1 \to C_2 \to \dots C_n \to q$ in Δ we have that for all i
$\Delta \vdash^? C_i, H * q$ succeeds.

Example 2.3.1 $\vdash^? ((p \to q) \to q) \to (q \to p) \to p, \emptyset,$
$(p \to q) \to q \vdash^? (q \to p) \to p, \emptyset,$
$(p \to q) \to q, q \to p \vdash p, \emptyset,$
$(p \to q) \to q, q \to p \vdash q, (p),$
$(p \to q) \to q, q \to p \vdash p \to q, (p, q),$
$(p \to q) \to q, q \to p, p \vdash q, \emptyset,$
$(p \to q) \to q, q \to p \vdash p \to q, (q),$
$(p \to q) \to q, q \to p \vdash q, (q),$
fail

■

In this way one is able to detect a loop (the same atomic goal repeats from the same database), without having to record each pair (database goal). We will use a similar idea to give a terminating procedure for the implicational fragment of relevance logic R in chapter 5.

However, loop-checking is not the only way to ensure termination. Trying to think constructively, in order to make sure the algorithm terminates, let us ask ourselves what is involved in the computation. There are two parameters; the data and the goal. The goal is reduced to an atom via the rule for $\to$ and the looping occurs because a data item is being used by the rule for reduction again and again. Our aim in the historical loop checking is to stop that. Well, why don't we try a modified rule for reduction which can use each item of data only once? This way, we will certainly run out of data and the computation will terminate. We have to give a formal definition of the computation, where we keep track on how many times we use the data.

Let us adopt the point of view that each database item can be used at most once. Thus our rule for reduction becomes

from $\Delta \ \vdash^? \ q$,
if there is $B \in \Delta$, with $B = C_1 \to C_2 \to \ldots C_n \to q$, step to
$\Delta - \{B\} \ \vdash^? \ C_i$ for $i = 1, \ldots, n$

The item B is thus thrown out as soon as it is used.

Let us call such a computation *locally linear computation* (bounded resource computation) where each formula can be used at most once in each path of the computation. That is why we are using the word 'locally'. One can also have the notion of (globally) linear computation, in which each formula can be used exactly once in the entire computation tree.

Before we proceed, we need to give a formal definition of these concepts. We give Definition 2.3.2 below and it should be compared with Definition 2.2.1 of the previous section.

Since we take care of usage of formulas, it is natural to regard multiple copies of the same formula as distinct. This means that databases can now be considered as multiset of formulas. In order to keep the notation simple, we use the same notation as in the previous section. From now on, Γ, Δ, etc. will range on multisets of formulas, and we will write Γ, Δ to denote the union multiset of Γ and Δ, that is $\Gamma \sqcup \Delta$. To denote a multiset $[A_1, \ldots A_n]$, if there is no risk of confusion we will simply write $A_1, \ldots A_n$ (see chapter 1, for formal definitions of multiset and relative notions).

Definition 2.3.2 [Goal directed computation, locally linear goal directed computation and linear goal directed computation] We present three notions of computation by defining their computation trees. These are:

1. The goal directed computation for intuitionistic logic.

2. Locally linear goal directed computation (LL computation).

3. Linear goal directed computation.

We give the computation rules for a query: $\Gamma \ \vdash^? \ G$, where Γ is a multiset of formulas and G is a formula.

- (success) $\Delta \ \vdash^? \ q$ immediately succeeds if the following holds:
    1. for intuitionistic and LL computation, $q \in \Delta$,
    2. for linear computation $\Delta = q$.
- (implication) From $\Delta \ \vdash^? \ A \to B$, we step to

    $\Delta, A \ \vdash^? \ B$.
- (reduction) If there is a clause $B \in \Delta$ with

$$B = C_1 \to \ldots \to C_n \to q$$

    then from $\Delta \ \vdash^? \ q$, we step, for $i = 1, \ldots, n$ to

    $\Delta_i \ \vdash^? \ C_i$,

    where the following holds

    1. in the case of intuitionistic computation, $\Delta_i = \Delta$;

2. in the case of locally linear computation, $\Delta_i = \Delta - [B]$;
3. in the case of linear computation $\sqcup_i \Delta_i = \Delta - [B]$.

■

COMMENT: I HAVE DELETED THE CONDITION $\Delta_i \cap \Delta_j = \emptyset$, SINCE THE INTERSECTION OF MULTISETS DOES NOT MAKE SENSE AND IT IS IMPLICIT IN THE CONDITION $\sqcup_i \Delta_i = \Delta - [B]$ THAT THE ELEMENT OF $\Delta - [B]$ ARE PARTITIONED IN THE Δ_i's.

The question is now: do we retain completeness? Are there examples for intuitionistic logic where items of data essentially need to be used locally more than once?

It is reasonable to assume that if things go wrong, they do so with formulas with at least two nested implications, i.e. formulae with the structure $X \to Y \to Z$ or $(X \to Y) \to Z$. Namely, by adding new atoms and renaming implicational subformulas by the new atoms, we can reduce any database to an equivalent database with only two levels of nested implication.

Sooner or later we will discover the following examples.

Example 2.3.3 1. $c \to a, (c \to a) \to c \vdash^? a$

The clause $(c \to a)$ has to be used twice in order for a to succeed. This example can be generalized.

2. Let $A_0 = c$
$A_{n+1} = (A_n \to a) \to c$.

Consider the following query:

$$A_n, c \to a \vdash^? a$$

The clause $c \to a$ has to be used $n+1$ times (locally).

■

Example 2.3.4 Another example in which a formula must be used locally more than once is the following: the database contains

$(b_1 \to a) \to c$
$\vdots$
$(b_n \to a) \to c$
$b_1 \to b_2 \to \ldots b_n \to c$
$c \to a$
the goal is 'a'.

It is easy to see that $c \to a$ has to be used $n+1$ times locally to ensure success. ■

Example 2.3.5

$$a \to b \to c, a \to b, a \vdash^? c$$

Let us do the full LL-computation:

Here a has to be used twice globally, but not locally on each branch of the computation. Thus the locally linear computation succeeds. On the other hand, in the linear computation case, this query fail, as a must be used globally twice. ■

$$\begin{tikzcd}
 & a \to b \to c, a \to b, a \vdash^? c \arrow[dl, dash] \arrow[dr, dash] & \\
a \to b, a \vdash^? a & & a \to b, a \vdash^? b \arrow[d, dash] \\
 & & a \vdash^? a
\end{tikzcd}$$

Figure 2.2:

The example above shows that that the locally linear proof system of 2.3.2 is not the same as the linear proof system. First we do not require that all assumptions must be used, condition on success rule, we only ask that they be used no more times than specified. A more serious difference is that we do our 'counting' of how many times a formula is used separately on each path of the computation and not globally for the entire computation. The counting in the linear is global, as can be seen by the condition in reduction rule.

Another example, the query $A, A \to A \to B \vdash^? B$ will succeed in our locally linear computation because A is used once on each of two parallel paths. It will not be accepted in linear computation because A is used globally twice. This is ensured by condition in reduction rule.

With respect to the notion of intuitionistic consequence relation introduced at the beginning of the chapter, we can notice that that linear computation does *not* does not satisfy all the properties involved in the definition. Monotonicity does not hold. Reflexivity holds in the restricted form $A \vdash A$. The cut rule holds in the form

$$\frac{\Delta \vdash A \quad \Gamma, A \vdash B}{\Delta, \Gamma \vdash B}$$

holds. On the other hand, as a difference with respect to the intuitionistic case, the cut rule does not hold in the form

$$\frac{\Delta \vdash A \; \Delta, A \vdash B}{\Delta \vdash B}$$

The two form of cut are easily seen to be equivalent in intuitionistic logic, whereas they are not for globally linear computation. A counterexample to the latter form is the following:

$a, a \vdash^? (a \to b) \to (a \to b \to c) \to c$ succeeds, and
$a \vdash^? a$ succeeds.

Therefore, by cut we should get

$a \vdash^? (a \to b) \to (a \to b \to c) \to c$ succeeds,

which, of course, does not hold. Linear computation as defined in 2.3.2 corresponds to linear logic implication, [Girard 87], in the sense that the the procedure of linear computation is sound and complete for the implicational fragment of linear logic, this will be proved in chapter 6, within the broader context of substructural logics. Linear logic has the implication connective $-\circ$ and $A_1, \ldots, A_n \vdash B$ in linear logic means that B can be proved from A_i using each A_i exactly once. A deduction theorem holds, namely that $A_1, \ldots, A_n \vdash B$ is equivalent to $\vdash A_1 - \circ(A_2 - \circ \ldots (A_n - \circ B) \ldots)$.

In case of LL computation (that is, locally linear computation), the properties of reflexivity and monotonicity are satisfied, but cut is not, here is a counterexample: letting $A = (a \to b) \to (a \to b) \to b$, we have

$A, a \to b \vdash^? b$ succeeds, and
$(a \to b) \to a \vdash^? A$ succeeds.

On the other hand, as we have seen

$(a \to b) \to a, a \to b \vdash^? b$ fails.

The above examples show that we do not have completeness with respect to intuitionistic provability for locally linear computations. Still the locally linear computation is atractive, because if the database is finite it is always terminating and it is efficient. It is natural to wonder whether we can compensate for the use of the locally linear rule (i.e. for throwing out the data) by some other means. Moreover, even if the locally linear computation is not complete for the full intuitionistic implicational fragment, one may still wonder whether it works in some particular and significant case. A significant case is shown in the next proposition.

Proposition 2.3.6 *The locally-linear procedure is complete for Horn databases.*

2.3.1 Bounded restart rule for intuitionistic logic

Let us now go back to the notion of locally linear computation. We have seen that the locally linear restriction does not retain completeness with respect to intuitionistic provability. There are examples where formulae need to be used locally several times. Can we compensate? Can we at the same time throw data out once it has been used and retain completeness for intuitionistic logic by adding some other computation rule? The answer is yes. The rule is called the *(linear) bounded restart rule* and is used in the context of the notion of locally linear computation with history.

Let us examine more closely why we needed in Example 2.3.3 the clause $c \to a$ several times. The reason was that from other clauses, we got the query '$\vdash^? a$' and we wanted to use $c \to a$ to continue to the query '$\vdash^? c$'. Why was not $c \to a$ available ?; because $c \to a$ has already been used. In other words, $\vdash^? a$ as a query, has already been asked and $c \to a$ was used. This means that the next query after '$\vdash^? a$' in the history was '$\vdash^? c$'.

If H is the history of the atomic queries asked, then somewhere in H there is '$\vdash^? a$' and immediately afterwards '$\vdash^? c$'.

We can therefore compensate for the re-use of $c \to a$ by allowing ourselves to go back in the history to where '$\vdash^? a$' was, and allow ourselves to ask all queries that come afterwards. Let us see what happens to our example 2.3.3

$(c \to a) \to c$	[1]	$\vdash^? \ a$
$c \to a$	[1]	

We use the second clause to get

$(c \to a) \to c$	[1]	$\vdash^? \ c, (a)$
$c \to a$	[0]	

we continue

$(c \to a) \to c$	[0]	$\vdash^? \ c \to a, (a, c)$
$(c \to a)$	[0]	

we continue

$(c \to a) \to c$	[0]	$\vdash^? \ a, (a, c)$
$c \to a$	[0]	
c	[1]	

The '1'('0') annotate the clause to indicate it is active (inactive) for use. The history can be seen as the right hand column of past queries.

We can now ask any query that comes after an 'a' in the history, hence

$(c \to a) \to c$	[0]	
$(c \to a)$	[0]	$\vdash^? \ c, (a, c, a)$
c	[1]	

Success.

The previous example suggests the following new computation with bounded restart rule. For technical reasons, from now on we regard again databases as *sets* of formulas rather than multisets. However, the results which follows hold indifferently for databases which are either sets or multiset of formulas. To this concern we observe that, whereas for the locally linear computation without bounded restart it makes a great difference whether the database is regarded as a set or as a multiset, once that bounded restart is added the difference is no longer significant.

Definition 2.3.7 [Locally linear computation with bounded restart] In the computation with bounded restart, the queries have the form $\Delta \ \vdash^? \ G, H$, where Δ is a set of formulas and the history H is a sequence of atomic goals. The rules are as follows.

- (success) $\Delta \ \vdash^? \ q, H$ succeeds if $q \in \Delta$.
- (implication) from $\Delta \ \vdash^? \ A \to B, H$ step to

  $\Delta, A \ \vdash^? \ B, H$
- (reduction) from $\Delta \ \vdash^? \ q, H$ if $C = D_1 \to D_2 \to \ldots \to D_n \to q \in \Delta$, then we step to

  $\Delta - \{C\} \ \vdash^? \ D_i, H * (q)$ for $i = 1, \ldots, n$.
- (Bounded restart) from $\Delta \ \vdash^? \ q, H$ step to

  $\Delta \ \vdash^? \ q_1, H * (q)$,

  provided for some H_1, H_2, H_3, it holds $H = H_1 * (q) * H_2 * (q_1) * H_3$, where each H_i may be empty.

■

Note that the databases involved in a computation need not be finite. In a successful computation tree only a finite number of clauses from the database will be used in the sense mentioned above.

Both soundness and completeness of this procedure with respect to intuitionistic provability may not be evident. Consider the following two databases:

$$\Delta_1 = \{(c \to a) \to b\} \text{ and } \Delta_2 = \{c, a \to b\}$$

If we ask the query $\Delta_1 \vdash^? b$ and $\Delta_2 \vdash^? b$, using the linear bounded restart computation, we get that both queries reduce to $c \vdash^? a, (b)$. Thus we see that one cannot reconstruct the original database from the history[1]. This problem has bearing upon soundness.

However, the procedure with bounded restart is sound and complete with respect to intuitionistic provability, and this is what we are going to show next.

Lemma 2.3.8 *The following hold in intuitionistic logic:*

1. *If* $\Delta, A, B \to C \vdash B$
   Then $\Delta, (A \to B) \to C \vdash A \to B$

2. $(A \to C) \to A, (B \to C) \to B \vdash (A \to B \to C) \to (A \wedge B)$.

Proof. Left to the reader. □

Theorem 2.3.9 (Soundness of locally linear computation with bounded restart) *For the computation of 2.3.7 we have that: if* $\Delta \vdash^? G, (p_1, \ldots, p_n)$ *succeeds then* $\Delta, G \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash G$ *holds.* [2]

Proof. By induction on the height h of a successful proof of (1). If $h = 0$ then $G = q \in \Delta$ and the result follows immediately. Let $h > 0$. Let $G = A \to B$. The computation steps to

$$\Delta, A \vdash^? B, (p_1, \ldots, p_n)$$

by induction hypothesis we have that

$$\Delta, A, B \to p_n, p_{n-1}, \ldots, p_2 \to p_1 \vdash B.$$

By lemma 2.3.8(1), we have that

$$\Delta, (A \to B) \to p_n, p_{n-1}, \ldots, p_2 \to p_1 \vdash A \to B.$$

Let G be an atom q. Suppose that reduction rule is applied to q, then $\Delta = \Delta', C$, with $C = D_1 \to \ldots \to D_k \to q$, and the computation steps for $i = 1, \ldots, k$ to

$$\Delta' \vdash^? D_i, (p_1, \ldots, p_n, q)$$

by induction hypothesis, we have that

$$\Delta', D_i \to q, q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash D_i,$$

by lemma 2.3.8(2), for $i = 1, \ldots, k$, we get

[1]A similar reduction is embodied in the contraction-free calculi for intuitionistic logic by Dickhoff [Dyckhoff 92] and by Hudelmaier [Hudelmaier 90] ADD REFERENCE TO JLC PAPER.

[2]This theorem is a correct version of theorem 5.0.8, pag. 351 of [Gabbay 92], which contained a mistake.

$$\Delta', D_1 \to \ldots \to D_k \to q, q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash D_i,$$

and hence,

$$\Delta', C, q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash q.$$

Finally, suppose bounded restart rule is applied to q. Then from

$$\Delta \ \vdash^? \ q, (p_1, \ldots, q \ldots, p_j, \ldots, p_n)$$

the computation steps to

$$\Delta \ \vdash^? \ p_j, (p_1, \ldots, q \ldots, p_j, \ldots, p_n, q)$$

that is, it must be $q = p_i$, with $i < j \leq n$. By induction hypothesis, we have that

$$\Delta, p_j \to q, q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash p_j,$$

but since $q = p_i$ precedes p_j, we have

$$q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash p_j \to q,$$

and hence we obtain

$$\Delta, q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash q.$$

$\square$

We now turn to completeness We first prove that if $\Delta \vdash G$ then $\Delta \ \vdash^? \ G, \emptyset$ succeeds.

To this purpose we introduce an intermediate procedure, we call it P_i. In P_i, data are regarded as sets and each formula can be used locally more than once, but after its first use it is removed from the database, say Δ, and recorded in a separate part, Γ, "the storage"; the data in the storage are listed according to usage. The storage Γ only contains non-atomic formulas and the sequence of their heads forms the history of the procedure with bounded restart. For technical reasons, we allow in P_i the rule of bounded restart in a restricted form. It should be clear that this rule is redundant. Queries for P_i have the form

$$\Delta \mid \Gamma \ \vdash^? \ G,$$

where Δ is a set of formulas and Γ is a list of non-atomic formulas. The rules of P_i are as follows:

- (success) $\Delta \mid \Gamma \ \vdash^? \ q$ succeeds if $q \in \Delta$;
- (implies) from $\Delta \mid \Gamma \ \vdash^? \ A \to B$ step to $\Delta \cup \{A\} \mid \Gamma \ \vdash^? \ B$;
- (reduction)if there is a formula $C = B_1 \to \ldots \to B_n \to q \in \Delta \cup \Gamma$, then from $\Delta \mid \Gamma \ \vdash^? \ q$ step to $\Delta' \mid \Gamma * C \ \vdash^? \ B_i$, for $i = 1, \ldots, n$, where $\Delta' = \Delta - \{C\}$ if $C \in \Delta$ and $\Delta' = \Delta$ if $C \in \Gamma$.
- (bounded restart) if $\Gamma = \Gamma_1 * C_1 * C_2 * \Gamma_2$, with $Head(C_1) = q$ and $Head(C_2) = r$, then from

  $\Delta \mid \Gamma \ \vdash^? \ q$ step to
  $\Delta \mid \Gamma * C_1 \ \vdash^? \ r$

As one can see, formulas of Δ are moved to Γ the first time they are used. In case C occurs in both Δ and Γ one can choose to apply reduction with respect to either one or the other occurrence of C and the database Δ' is determined accordingly. Notice that in P_i, the storage Γ plays a double role, as history for bounded restart and as database for reduction steps. We will show how to simulate reduction steps with respect the same formula along one branch (that is reduction steps with respect to formulas in the storage) by bounded restart steps.

One can easily prove that P_i is equivalent to the procedure for intuitionistic provability.

Lemma 2.3.10 *A query $\Delta \vdash^? G$ succeeds in the basic intuitionistic procedure iff the query $\Delta \mid \emptyset \vdash^? G$ succeeds in P_i.*

We need the following lemma. Let $Atom(\Sigma)$, $N\_Atom(\Sigma)$ respectively denote the set of atomic and non-atomic formulas in Σ.

Lemma 2.3.11 *If $\Delta \cup \Sigma \mid \Gamma \vdash^? G$ succeeds in P_i and $Atom(\Sigma) \subseteq \Delta$, $N\_Atom(\Sigma) \subseteq \Delta \cup \Gamma$, then the query $Q = \Delta \mid \Gamma \vdash^? G$ succeeds in P_i by a derivation of no greater size (number of nodes).*

Proof. The proof of this lemma is by induction on the height of a successful derivation of the query of the hypothesis. The cases of (success), (implication) and (bounded restart) are straightforward. We only consider the relevant case of reduction. Suppose, from $\Delta \cup \Sigma \mid \Gamma \vdash^? q$ we step to $Q_i = (\Delta \cup \Sigma)' \mid \Gamma * C \vdash^? B_i$, for $i = 1, \ldots, n$, for some $C = B_1 \rightarrow \ldots \rightarrow B_n \rightarrow q \in \Delta \cup \Sigma \cup \Gamma$, where the precise form of $(\Delta \cup \Sigma)'$ depends on which database contains C.We have several cases. If $C \in \Delta$, we step to $(\Delta - \{C\}) \cup (\Sigma - \{C\}) \mid \Gamma * C \vdash^? B_i$, It is obvious that $\Sigma - \{C\}$ satisfies the hypothesis of the lemma, hence we can apply the induction hypothesis, that is for $i = 1, \ldots, n$, $Q'_i = \Delta - \{C\} \mid \Gamma * C \vdash^? B_i$ succeeds, so that we can apply reduction wrt. $C \in \Delta$ and step, from $\Delta \mid \Gamma \vdash^? q$, to Q_i and succeed. If $C \in \Gamma$, we proceed as before by applying the induction hypothesis and performing a reduction step w.r.t. $C \in \Gamma$. Finally, if $C \in \Sigma - \Delta$, then $C \in \Gamma$, and we are back to the previous case. Since there is one-to-one mapping it is clear that the size of the derivation of Q is not greater than that of the original query. □

We are now in the position of proving the completeness of bounded restart computation. We show that every reduction step with respect to a formula of Γ can be simulated by a bounded restart step. Let $\mathcal{D}$ be a derivation of a query $Q_0 = \Delta_0 \mid \emptyset \vdash^? G_0$. Given a query $Q = \Delta \mid \Gamma \vdash^? q$, we say that a reduction step is "bad" if it is performed with respect to a formula of Γ. A derivation without "bad" reduction steps is a derivation according to the procedure 2.3.7, with the small difference that in the latter we just record the head of the formulas of the storage part Γ.

Lemma 2.3.12 *If $Q_0 = \Delta_0 \mid \emptyset \vdash^? G_0$ succeeds by a derivation in P_i, then it succeeds by a derivation which does not contain "bad" reduction steps, that is a locally linear derivation with bounded restart.*

Proof. Let $\mathcal{D}_0$ be a P_i-derivation of Q_0. Working from the root downwards, we replace every "bad" reduction step by a bounded restart step. Let $Q = \Delta \mid \Gamma \vdash^? q$ be a query in $\mathcal{D}_0$, suppose that the successors of Q in $\mathcal{D}_0$ are obtained by a bad reduction step, that is a reduction step with respect to a formula $C_1 \in \Gamma$; let $\Gamma = \Gamma_1 * C_1 * \Gamma_2$ and $C_1 = B_1 \rightarrow \ldots \rightarrow B_n \rightarrow q$, then the successors of Q are

$$Q_i = \Delta \mid \Gamma * C_1 \vdash^? B_i, \text{ for } i = 1, \ldots, n.$$

Since $C_1 \in \Gamma$, C_1 must have already been used at a previous step on the same branch, that is there is a query $Q' = \Delta' \mid \Gamma_1 \vdash^? q$, and for some i, say $i = 1$, there is a successor of Q'

$$Q_1' = \Delta' \mid \Gamma_1 * C_1 \vdash^? B_1,$$

on the same branch. In the most general case, let $B_1 = D_1 \to \ldots \to D_k \to r$, with $r > 0$ and $\Sigma_1 = \{D_1, \ldots, D_k\}$. Then, under Q_1' on the same branch there is a query

$$Q_1^* = \Delta' \cup \Sigma_1 \mid \Gamma_1 * C_1 \vdash^? r.$$

It is clear that in case B_1 is atomic, $\Sigma_1 = \emptyset$ and $Q_1^* = Q_1'$. We can assume that Q is a descendent of Q_1^*. If it were $Q_1^* = Q$, then $q = r$ and the reduction step performed on Q would just be an immediate repetition of the reduction step performed on Q'. We can assume that $\mathcal{D}_0$ does not contain such immediate repetitions. The query Q_1 will have a descendant corresponding to Q_1^*, that is

$$Q_1^{**} = \Delta \cup \Sigma_1 \mid \Gamma * C_1 \vdash^? r.$$

To sum up, the sequence of the queries along the branch (from the root downwards) is as follows: Q', Q_1', Q_1^*, Q, Q_1, Q_1^{**}, where it might be $Q_1' = Q_1^*$ and $Q_1 = Q_1^{**}$. Since Q is a descendent of Q_1^*, it must be $\Gamma = \Gamma_1 * C_1 * \Gamma_2$, with $\Gamma_2 \neq \emptyset$, that is $\Gamma_2 = C_2 * \Gamma_2'$ and $Head(C_2) = r$, no matter whether the successors of Q_1^* are determined by reduction or bounded restart. Moreover, since Q follows Q_1^*, $Atom(\Sigma_1) \subseteq \Delta$, and $N\_Atom(\Sigma_1) \subseteq \Delta \cup \Gamma$. Thus, we can apply the previous lemma so that we have that

$$Q_1'' = \Delta \mid \Gamma * C_1 \vdash^? r.$$

succeeds by a derivation, say $\mathcal{D}_1^*$ of size no greater than the size of the subderivation $\mathcal{D}_1$ of Q_1^{**}. Now, we can replace the subderivation with root Q (and successors) Q_i, by the following:

$$\begin{array}{c} Q \\ \downarrow \\ Q_1'' \\ \mathcal{D}_1^* \end{array}$$

and we can justify the step from Q to Q_1'' by bounded restart. We have described one transformation step. Proceeding downwards from the root, each transformation step reduces by one the number of bad reductions at maximal level (closest to the root), without increasing the derivation size (the number of nodes), thus the entire process must terminate and the final derivation does not contain bad reductions.

□

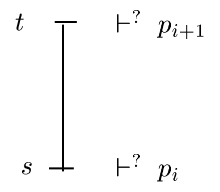
Theorem 2.3.13 (Completeness of locally linear computation with bounded restart) *For the computation of 2.3.7 we have that: if $\Delta, G \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash G$ holds, then $\Delta \vdash^? G, (p_1, \ldots, p_n)$ succeeds.*

Proof. It is sufficient, in view of the deduction theorem, to assume that $G = q$ is atomic. Let $\Delta' = \Delta \cup \{q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1\}$ then if $\Delta' \vdash q$ then $\Delta \vdash^? q, (p_1, \ldots, p_n)$ succeeds .

By the previous lemma, we have that

$$\Delta, q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash^? q, \emptyset \text{ succeeds.}$$

Let $\mathcal{D}$ be a derivation of the above query. For each node $t \in \mathcal{D}$ let $Q_t = \Delta_t, \vdash^? G_t, H_t$. Let $\mathbf{X}$ be the sequence $(p_1, \ldots, p_n)$. Let $H_t' = \mathbf{X} * H_t$. Let $Q_t' = \Delta_t, \vdash^? G_t, H_t'$. Let $\mathcal{D}'$ be a derivation for $\Delta \cup \{q \to p_n, \ldots, p_2 \to p_1\} \vdash^? q, \mathbf{X}$, obtained by replacing each Q_t by Q_t'. All uses of the bounded restart rule in $\mathcal{D}'$ are still valid because we have appended $\mathbf{X}$ at the beginning of H_t. Moreover we can use the bounded restart rule to perform any step of the form



rather than using the formulas since $\mathbf{X}$ is appended at the beginning of H'_t. Thus, $\mathcal{D}'$ is also a successful derivation of $\Delta, \{q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1\} \ \vdash^? \ q, \mathbf{X}$. In $\mathcal{D}'$, the clauses of $\{q \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1\}$ are *never* used. Thus these can be taken out of the database and $\mathcal{D}'$ is a successful derivation of $\Delta \ \vdash^? \ q, \mathbf{X}$ succeeds. □

2.3.2 Restart rule for classical Logic

We now introduce a new rule, the restart rule. It is a variation of the bounded restart rule obtained by cancelling any restrictions and simply allowing us to ask any earlier atomic goal. We need not keep history as a sequence but only as a *set* of atomic goals. The rule becomes

Definition 2.3.14 [Restart rule in the LL-computation] If $a \in H$, from $\Delta \ \vdash^? \ q, H$ step to $\Delta \ \vdash^? \ a, H \cup \{q\}$.

The formal definition of *locally linear computation with restart* is Definition 2.3.7 with the additional restart rule above in place of the bounded restart rule. ■

Example 2.3.15

$$(a \to b) \to a \ \vdash^? \ a, \emptyset$$

use the clause and throw it out to get:

$$\emptyset \ \vdash^? \ a \to b, \{a\}$$

and

$$a \ \vdash^? \ b, \{a\}$$

restart

$$a \ \vdash^? \ a, \{a, b\}$$

success.

The above query fails in the intuitionistic computation. Thus, this example shows that we are getting a logic which is stronger than intuitionistic logic. Namely, we are getting classical logic. This claim has to be properly proved, of course. ■

If we adopt the basic computation procedure for intuitionistic implication 2.2.1 rather tan the LL-computation, we can restrict the restart rule to always choose the *initial goal* as the goal with which we restart. Thus, we do not need to keep the history, but only the initial goal and the rule becomes more deterministic. On the other hand, the price we pay is that we cannot throw out the formulas of database when they are used.

Definition 2.3.16 [Simple computation with restart] We have that queries have the form

$$\Delta \vdash^? G, (G_0),$$

where G_0 is a goal. The computation rules are the same as in the basic computation procedure for intuitionistic logic 2.2.1 plus the following rule

(Restart) from

$$\Delta \vdash^? q, (G_0)$$

step to

$$\Delta \vdash^? G_0, (G_0).$$

It is clear that the initial query of any derivation will have the form $\Gamma \vdash^? A, (A)$. ■

It is natural to wonder what we get if we add the restart rule of definition 2.3.14 to the basic procedure for intuitionistic logic of 2.2.1. In the next lemma we show that given the underlying computation procedure of 2.2.1, restarting from an arbitrary atomic goal in the history is equivalent to restart from the initial goal. To this purpose, let $\vdash^?_{RI}$ and $\vdash^?_{RA}$ be respectively the deduction procedure of 2.3.16 and the deduction procedure of 2.2.1 extended by the restart rule of definition 2.3.14.

Lemma 2.3.17 *For any database Δ and formula G, we have*

(1) $\Delta \vdash^?_{RA} G, \emptyset$ succeeds iff (2) $\Delta \vdash^?_{RI} G, (G)$ succeeds.

Proof. Let $G = A_1 \to \dots A_k \to q$, with $k \geq 0$ and let $\Delta' = \Delta \cup \{A_1, \dots, A_k\}$. By (1) we have

(1)$\Delta \vdash^?_{RA} G, \emptyset$ succeeds iff (1') $\Delta' \vdash^?_{RA} q, \emptyset$ succeeds.

On the other hand, by (2) and the fact that Δ' is a set and it never decreases, we have

(2) $\Delta \vdash^?_{RI} G, (G)$ succeeds iff (2') $\Delta' \vdash^?_{RI} q, (G)$ succeeds iff $\Delta' \vdash^?_{RI} q, (q)$ succeeds.

($\Leftarrow$) Unless $q \in \Delta'$, in such a case both (1) and (2) immediately succeed, (2') will step by reduction to some query $\Delta' \vdash^?_{RI} A_i, (q)$; we can perform the same reduction step from (1'), and the resulting queries will contain q as the first element of the history and we are done.

($\Rightarrow$) Let (1') succeeds, consider a successful derivation of (1'). Suppose restart rule is applied to a query Q_0 by re-asking an atomic goal p which is the goal of an ancestor query Q of Q_0, it suffices to show that we still obtain a successful derivation if we restart by re-asking an atomic goal which comes from any ancestor query Q' of Q. This fact implies what we have to prove, namely it is more general. To show the fact above, suppose the following situation occurs: in one branch of a given derivation $\mathcal{D}$ we have

$$(1)\ \Delta_1 \vdash^?_{RA} q_1, H_1$$

$$\vdots$$

$$(2)\ \Delta_2 \vdash^?_{RA} q_2, H_2$$

$$\vdots$$

$$(3)\ \Delta_3 \vdash^?_{RA} q_3, H_3$$

The branch goes on by restart using $q_2 \in H_3$, that is we step to

$$(4)\ \Delta_3 \ \vdash^?_{RA}\ q_2, H_3 \cup \{q_3\}.$$

Since $q_1 \in H_2 \subseteq H_3$, we can build an alternative derivation $\mathcal{D}'$, which coincides with $\mathcal{D}$ up to (3) and goes on by restart using q_1; that is after (3) we step to

$$(4')\ \Delta_3 \ \vdash^?_{RA}\ q_1, H_3 \cup \{q_3\}.$$

Since $\Delta_1 \subseteq \Delta_2 \subseteq \Delta_3$, by the monotony property, from (4') we can reach (4), by the same steps we have done to reach (2) from (1). □

As it is clear from the proof of the above lemma, we can further restrict restart from initial goal G_0 to restart from the first initial goal $q_0 = \mathit{Head}(G_0)$.

In the following we prove completeness with respect to classical logic of two proof procedures: one is the basic procedure for intuitionistic logic with the additional rule of restart from initial goal, and the other one is the locally linear-computation procedure where the databases are considered as sets of formulas, with the additional rule of restart from any previous atomic goal.

Soundness and Completeness of Restart from the initial goal

We show that the proof-procedure obtained by adding the rule of restart from the initial goal to the basic procedure for intuitionistic logic defined in 2.2.1 is sound and complete with respect to classical provability.

We define the notion of complement of a goal which works as its negation.

Definition 2.3.18 Let A be any formula. Then the complement of A, denoted by $\mathit{Cop}(A)$ is the following set of formulas:

$$\mathit{Cop}(A) = \{A \to p \mid p \text{ any atom of the language}\}$$

■

We show that we can replace any application of the restart rule by a reduction step using a formula in $\mathit{Cop}(A)$.

Lemma 2.3.19 *(1) $\Gamma \ \vdash^?\ A$ succeeds by using restart rule iff (2) $\Gamma \cup \mathit{Cop}(A)\ \vdash^?\ A$ succeeds with out using restart, that is by the intuitionistic procedure 2.2.1.*

Proof. We can easily establish a mapping between derivations of (1) and derivations of (2), each query $\Sigma\ \vdash^?\ G, (A)$ corresponds to a query $\mathit{Cop}(A) \cup \Sigma\ \vdash^?\ G$ and vice-versa: let $\Sigma\ \vdash^?\ r, (A)$ be any step in a derivation of (1), where restart is used, so that the next step will be $\Sigma\ \vdash^?\ A, (A)$, since $A \to r \in \mathit{Cop}(A)$, from the corresponding query $\mathit{Cop}(A) \cup \Sigma\ \vdash^?\ q$, we can step to $\mathit{Cop}(A) \cup \Sigma\ \vdash^?\ A$, by reduction with respect to $A \to r$. On the other hand, whenever we step from $\mathit{Cop}(A) \cup \Sigma\ \vdash^?\ q$ to $\mathit{Cop}(A) \cup \Sigma\ \vdash^?\ A$ by reduction with respect to $A \to q \in \mathit{Cop}(A)$, we can achieve the same result by restarting from A. □

We will prove first that $\Delta \vdash A$ in classical implicational logic iff $\Delta \cup \mathit{Cop}(A)\ \vdash^?\ A$ succeeds by the procedure defined in 2.2.1.

Lemma 2.3.20 *For any database Δ and formulas G such that $\Delta \supseteq Cop(G)$, and for any goal A, conditions (a) and (b) below imply condition (c):*

(a) $\Delta \cup \{A\} \vdash^? G$ *succeeds*
(b) $\Delta \cup Cop(A) \vdash^? G$ *succeeds*
(c) $\Delta \vdash^? G$ *succeeds.*

Proof. Since $\Delta \cup Cop(A) \vdash^? G$ succeeds and computations are finite, only a finite number of the elements of $Cop(A)$ are used in the computation. Assume then that

(b1) $\Delta, A \to p_1, \ldots, A \to p_n \vdash^? G$ succeeds.

We use the cut rule. Since $\Delta \supseteq Cop(G)$ we have $G \to p_i \in \Delta$ and hence by the computation rules $\Delta \cup \{A\} \vdash^? p_i$ succeeds . Now by cut on (b1) we get $\Delta \vdash^? G$ succeeds . □

Theorem 2.3.21 *For any Δ and any A, (a) is equivalent to (b) below:*

(a) $\Delta \vdash A$ *in classical logic*

(b) $\Delta \cup Cop(A) \vdash^? A$ *succeeds by the intuitionistic procedure defined in 2.2.1*

Proof.

1. Show (b) implies (a):
   Assume $\Delta \cup Cop(A) \vdash^? A$ succeeds Then by the soundness of our computation procedure we get that $\Delta \cup Cop(A) \vdash A$ in intuitionistic logic, and hence in classical logic. Since the *proof* is *finite* there is a finite set of the form $\{A \to p_i, \ldots, A \to p_n\}$ such that

   (a1) $\Delta, A \to p_1, \ldots A \to p_n \vdash A$ (in intuitionistic logic).

   We must also have that $\Delta \vdash A$, in classical logic, because if there were an assignment h making Δ true and A false, it would also make $A \to p_i$ all true, contradicting (a1).

   The above concludes the proof that (b) implies (a).

2. Show that (a) implies (b).
   We prove that if $\Delta \cup Cop(A) \vdash^? A$ does not succeed then $\Delta \nvdash A$ in classical logic. Let $\Delta_0 = \Delta \cup Cop(A)$. We define a sequence of databases $\Delta_n, n = 1, 2 \ldots$ as follows:
   Let $B_1, B_2, B_3, \ldots$ be an enumeration of all formulas of the language.
   Assume Δ_{n-1} has been defined and assume that $\Delta_{n-1} \vdash^? A$ does not succeed. We define Δ_n:
   If $\Delta_{n-1} \cup \{B_n\} \vdash^? A does not succeed$, let $\Delta_n = \Delta_{n-1} \cup \{B_n\}$. Otherwise from lemma 2.3.20 we must have:
   $\Delta_{n-1} \cup$ Cop $(B_n) \vdash^? A$ does not succeed.
   and so let $\Delta_n = \Delta_{n-1} \cup Cop(B_n)$.
   Let $\Delta' = \bigcup_n \Delta_n$
   Clearly $\Delta' \vdash^? A$ does not succeed.

Define an assignment of truth values h on the atoms of the language by $h(p) = \mathbf{true}$ iff $\Delta' \vdash^? p$ succeeds . We now prove that

for any $B, h(B) = \mathbf{true}$ iff $\Delta' \vdash^? B$ succeeds,

by induction on B.

(a) For atoms this is the definition.

(b) Let $B = C \to D$. We prove the two directions by simultaneous induction.

(b1) Suppose $\Delta' \vdash^? C \to D$ succeeds. If $h(C) = \mathbf{false}$, then $h(C \to D) = \mathbf{true}$ and we are done. Thus, assume $h(C) = \mathbf{true}$. By induction hypothesis, it follows that $\Delta \vdash^? C$ succeeds. Since, by hypothesis we have that $\Delta, C \vdash^? D$ succeeds, by cut we obtain that $\Delta \vdash^? D$ succeeds, and hence by induction hypothesis $h(D) = \mathbf{true}$.

(b2) If $\Delta' \vdash^? C \to q$ does not succeed, we will show that $h(C \to D) = \mathbf{false}$. Let $Head(D) = q$. we get

(1) $\Delta \vdash^? D$ does not succeed
(2) $\Delta, C \vdash^? q$ does not succeed.

Hence by induction hypothesis on (1) we have that $h(D) = \mathbf{false}$. We show that $\Delta' \vdash^? C$ must succeed. Suppose on the contrary that $\Delta' \vdash^? C$ does not succeed. Hence $C \notin \Delta'$. Let $B_n = C$ in the given enumeration. Since $B_n \notin \Delta_n$, by construction, it must be $Cop(C) \subseteq \Delta'$. In particular $C \to q \in \Delta'$, and hence $\Delta, C \vdash^? q$ succeeds, against (2). We have shown that $\Delta' \vdash^? C$ succeeds, whence $h(C) = \mathbf{true}$, by induction hypothesis. Since $h(C) = \mathbf{false}$, we obtain $h(C \to D) = \mathbf{false}$.

We can now complete the proof. Since $\Delta' \vdash^? A$ does not succeed, we get $h(A) = \mathbf{false}$. On the other hand, for any $B \in \Delta \cup Cop(A)$, $h(B) = \mathbf{true}$ (since $\Delta \cup Cop(A) \subseteq \Delta$) and $h(A) = \mathbf{false}$. This means that $\Delta \cup Cop(A) \not\vdash A$ in classical logic. This complete the proof.

□

From the above theorem and lemma 2.3.19 we immediately obtain completeness of the proof procedure with restart from the initial goal.

Theorem 2.3.22 *$\Delta \vdash A$ in classical implicational logic iff $\Delta \vdash^? A, (A)$ succeeds using the restart rule from the initial goal, added to the procedure of 2.2.1.*

Soundness and Completeness of the LL-procedure with restart from any previous goal

We now examine completeness for locally linear computation defined in 2.3.7 with the restart rule from any previous goal. As the next example shows, we cannot restrict the application of restart to the first atomic goal occurred in the computation.

Example 2.3.23 The following query succeeds:

$(p \to q) \to p, p \to r \vdash^? r, \emptyset,$

by the following computation:

$$\begin{array}{rcl}
(p \to q) \to p, p \to r & \vdash^? & r,\ \emptyset, \\
(p \to q) \to p & \vdash^? & p, \{r\}, \\
 & \vdash^? & p \to q,\ \{r, p\}, \\
p & \vdash^? & q,\ \{r, p\}, \\
p & \vdash^? & p,\ \{r, p\},\ \text{ restart from } p.
\end{array}$$

Ir is clear that restarting from r, the first atomic goal would not help. ■

Theorem 2.3.24 *[Soundness and completeness of locally linear computation with restart]*
$\Delta \vdash^? G, H$ succeeds iff $\Delta \vdash G \vee \bigvee H$ in classical logic.

Proof. *Soundness.* We prove soundness by induction on the length of the computation.

1. *Length 0*
   In this case $G = q$ is atomic and $q \in \Delta$.
   Thus $\Delta \vdash q \vee \bigvee H$.

2. *Length $k + 1$*

   (a) $G = A \to B$
   $\Delta \vdash^? G, H$ succeeds if $\Delta \cup \{A\} \vdash^? B, H$ succeeds and hence $\Delta \cup \{A\} \vdash B \vee \bigvee H$ by the induction hypothesis and hence $\Delta \vdash G \vee \bigvee H$.

   (b) $G = q$ and for some $B = B_1 \to \ldots \to B_n \to q \in \Delta$, $(\Delta - \{B\}) \vdash^? B_i, H \cup \{q\}$ succeeds for $i = 1, \ldots, n$.
   By the induction hypothesis $\Delta - \{B\} \vdash B_i \vee q \vee \bigvee H$ for $i = 1 \ldots n$.
   However in classical logic $\bigwedge_{i=1}^{n}(B_i \vee q) \equiv (B_1 \to \ldots \to B_n \to q) \to q$. Hence $\Delta - \{B\} \vdash (B \to q) \vee \bigvee H$ and by the deduction theorem $\Delta \vdash q \vee \bigvee H$.

   (c) The restart rule was used, i.e. for some $a \in H$

   $$\Delta \vdash^? a, H \cup \{q\} \text{ succeeds.}$$

   Hence $\Delta \vdash a \vee q \vee \bigvee H$ and since $a \in H$ we get $\Delta \vdash q \vee \bigvee H$

I FINALLY FOUND A FORM OF INDUCTION WHICH WORKS, IN THE PREVIOUS VERSION IT DID NOT WORK.

Completeness. We prove completeness by induction on the complexity of the query, defined as follows.

Let $Q = \Delta \vdash^? G, H$, we define $cp(Q)$ as the *multiset* of the complexity values of non-atomic formulas in $\Delta \cup \{G\}$ if any, and the empty multiset otherwise, i.e.

$$cp(Q) = \begin{cases} [cp(A_1), \ldots, cp(A_n)] \text{ if } N\_Atom(\Delta \cup \{G\}) = \{A_1, \ldots, A_n\} \\ \emptyset \text{ if } N\_Atom(\Delta \cup \{G\}) = \emptyset \end{cases}$$

We now consider the following relation on integer multisets: given $\alpha = [n_1, \ldots, n_p]$ and $\beta = [m_1, \ldots, m_q]$, we write $\alpha \prec \beta$ if the following holds:

either $\alpha \subset \beta$, or α is obtained from β by replacing some occurrence of $m_i \in \beta$ by a multiset of numbers strictly smaller than m_i.

It may be proved that the relation $\prec$ is a well-order on integer multisets (REFERENCES??? I THINK DERSHOWITZ-MANNA []).

We are ready for the completeness proof. Let $Q = \Delta \ \vdash^? \ G, H$, we show that if $\Delta \vdash G \vee \bigvee H$ in classical logic then Q succeeds by induction on $cp(Q)$.

1. The base of the induction, $cp(Q) = \emptyset$, occurs when G and the database Δ are all atoms; this case is clear, because it must be $\Delta \cap (\{G\} \cup H) \neq \emptyset$.

2. Let $cp(Q) \neq \emptyset$. If G is of the form $A \to B$, then we have

   (*) $\Delta \vdash (A \to B) \vee \bigvee H$ iff $\Delta, A \vdash B \vee \bigvee H$.

   Let Q' be the query $\Delta, A \ \vdash^? \ B, H$; since $cp(A), cp(B) < cp(A \to B)$, we easily obtain that $cp(Q') \prec cp(Q)$; thus by (*) and by induction hypothesis, we obtain that Q' succeeds, whence Q succeed by the implication rule.

3. Let $cp(Q) \neq \emptyset$ and let G be an atom q. If $\Delta \vdash q \vee \bigvee H$ there must be a formula $C \in \Delta$ such that $Head(C) \in \{q\} \cup H$, otherwise we can define a countermodel (by making false all atoms in $H \cup \{q\}$ and true all heads of formulas of Δ). Assume C is such a formula and $Head(C) = p_1$. If $Body(C)$ is empty then the computation succeeds either immediately, or by the restart rule (according to $p_1 = q$ or $p_1 \in H$). Otherwise, let $C = A_1 \to \ldots \to A_n \to p_1$, with $n > 0$. Then in classical logic we have

   (**) $\Delta \vdash q \vee \bigvee H$ iff $\Delta - \{C\} \vdash A_i \vee q \vee \bigvee H$, for $i = 1, \ldots, n$.

   Let $Q_i = \Delta - \{C\} \ \vdash^? \ A_i, H \cup \{q\}$, for $i = 1, \ldots, n$. Since $cp(C) = \Sigma_{i=1}^{n} cp(A_i) + n$, we have $cp(A_i) < cp(C)$. So that each $cp(Q_i) \prec cp(Q)$. By hypothesis, (**) and the induction hypothesis, each Q_i succeeds. We can now show that Q succeeds as follows. First, if $p_1 \neq q$ we use restart with $p_1 \in H$ and ask

   $$\Delta \ \vdash^? \ p_1, H \cup \{q\}$$

   Then, in any case, we proceed by reduction with respect to C and ask Q_i, for $i = 1, \ldots, n$, which succeed by the induction hypothesis.

□

2.3.3 Some efficiency consideration

The procedures based on locally linear computation are a good starting point for designing efficient automated-deduction procedures; on the one hand proof search is guided by the goal, on the other hand derivations have a smaller size since when a formula has to be reused does not create further branching. We want now to remark upon termination of the procedures. The basic LL procedure obviously terminate, since formulas are thrown out as soon as they are used in a reduction step, every branch of a given derivation eventually ends with a query which either immediately succeeds, or no further reduction step is possible from it. This indeed was the starting motivation of the LL procedure as an alternative to a loop-checking mechanism. Does the (bounded) restart rule preserve this property? As we have stated it, it does not, in the sense that a silly kind of loop may be created by restart. Let us consider the following example, we give here the computation for intuitionistic logic, but the example works as well for the classical case:

$$\begin{array}{rcl} a \to b, b \to a & \vdash^? & a, \emptyset \\ a \to b & \vdash^? & b, (a) \\ & \vdash^? & a, (a, b) \\ & \vdash^? & b, (a, b, a) \\ & \vdash^? & a, (a, b, a, b) \\ & \vdots & \end{array}$$

This is a loop created by restart. It is clear that continuing the derivation by restart does not help, as no one of the new atomic goal matches the head of any formula in the database.

In case of classical logic, we can hence modify the restart rules as follows. From

$\Delta \ \vdash^? \ q, H,$

step to

$\Delta \ \vdash^? \ q_1, H \cup \{q\}$, provided there exists a formula $C \in \Delta$, with $q_1 = Head(C)$, and $q \in H$.

It is obvious that this restriction preserves completeness.

In case of intuitionistic logic, the situation is slightly more complex. The requirement that the atom from which we restart must match the head of some formula is too strong as the next example shows.

Example 2.3.25 Let Δ contain the following formulas:

$A_1 = s \to r,$
$A_2 = q \to s,$
$A_3 = (b \to r) \to q,$
$A_4 = b \to a \to r,$
$A_5 = (s \to q) \to a.$

Then, we have the following deduction:

$$\begin{array}{rcl} \Delta & \vdash^? & r, \ \emptyset \\ A_2, A_3, A_4, A_5 & \vdash^? & s, \ (r), \\ A_3, A_4, A_5 & \vdash^? & q, \ (r, s), \\ A_3, A_4, A_5 & \vdash^? & b \to r, \ (r, s, q), \\ A_4, A_5, b & \vdash^? & r, \ (r, s, q), \end{array}$$

Then we get

$A_5, b \vdash^? \ b, \ (r, s, q, r)$ and $A_5, b \vdash^? \ a, \ (r, s, q, r).$

The first query succeeds, the latter is reduced to

$b \vdash^? s \to q, (r, s, q, r, a),$
$b, s \vdash^? q, (r, s, q, r, a),$

we then apply bounded restart, the only options are r and a, the latter would not help, so that we choose r even if it does not match the head of any formula in the database

$b, s \vdash^? r, (r, s, q, r, a, q),$

we now apply bounded restart on s,

$b, s \vdash^? s, (r, s, q, r, a, q, r)$
success.

■

It should be clear that the atom with which we finally restart must match the head of some formula of the database in order to make any progress. But this atom might be reachable only through a sequence of restart steps which goes further and further back in the history. To handle this situation, we require that the atom chosen for restart matches some head, but we "collapse" several restart steps into a single one. In other words, we allow restart from a previous goal q which is accessible from the current one through a sequence of bounded restart steps.

Given a history $H = (q_1, q_2, \ldots, x_n)$, we define the relation "q_j is accessible from q_i in H", for q_j in the sequence, denoted by $Acc(H, q_i, q_j)$, as follows:

$$Acc(H, q_i, q_j) \equiv \exists q_k \in H\ (k < j \wedge q_k = q_i) \vee \exists q_k \in H(Acc(H, q_i, q_k) \wedge Acc(H, q_k, q_j)).$$

The modified bounded restart rule for intuitionistic logic becomes: from

$\Delta \vdash^? q, H,$

step to

$\Delta \vdash^? q', H * q$, provided

1. there exists a formula $C \in \Delta$, with $q' = Head(C)$, and
2. $Acc(H, q, q')$ holds.

It is easy to see that the above rule ensures the termination of the procedure without spoiling its completeness.

Another issue which is important from the point of view of proof search is whether backtracking is necessary or not when we are searching a derivation. The next lemmas shows that in case of classical logic, backtracking is not necessary, that is, it does not matter which formula we choose to match an atomic goal in a reduction step.

Lemma 2.3.26 *Let*

$$A = A_1 \to A_2 \to \ldots \to A_n \to q$$

and

$$B = B_1 \to B_2 \to \ldots \to B_m \to q.$$

Then (a) is equivalent to (b):

(a) $\Delta, A \ \vdash^? \ B_i, H \cup \{q\}$ *succeeds for* $i = 1, \ldots m$.

(b) $\Delta, B \ \vdash^? \ A_i, H \cup \{q\})$ *succeeds for* $i = 1, \ldots, n$.

Proof. By theorem 2.3.24, conditions (a) and (b) are equivalent, respectively, to (a') and (b') below:

(a') $\Delta, A \vdash B_i \vee q \vee \bigvee H$, for $i = 1, \ldots, n$,
(b') $\Delta, B \vdash A_i \vee q \vee \bigvee H$, for $i = 1, \ldots, n$.

By classical logic (a') and (b') are equivalent to (a'') and (b'') respectively:

(a'') $\Delta, A \vdash (\bigwedge_{i=1}^{m} B_i) \vee q \vee \bigvee H$
(b'') $\Delta, B \vdash (\bigwedge_{i=1}^{n} A_i) \vee q \vee \bigvee H$

Which are equivalent to (a''') and (b''') respectively:

(a''') $\Delta, A \vdash ((\bigwedge_{i=1}^{m} B_i \rightarrow q) \rightarrow q) \vee \bigvee H$
(b''') $\Delta, B \vdash ((\bigwedge_{i=1}^{n} A_i \rightarrow q) \rightarrow q) \vee \bigvee H$

Both (a''') and (b''') are equivalent to (c) below by the deduction theorem for classical logic.

(c) $\Delta, A, B \vdash q \vee \bigvee H$

This concludes the proof 2.3.26. □

By the previous lemma we immediately have.

Proposition 2.3.27 *In the computation* $\Delta \ \vdash^? \ q, H$ *with restart, no backtracking is necessary. The atom* q *can match with the head of any* $A_1 \rightarrow \ldots \rightarrow A_n \rightarrow q \in \Delta$ *and success or failure does not depend on the choice of such a formula.*

The parallel property to 2.3.26, 2.3.27 does not hold for the intuitionistic case. Let $A = A_1 \rightarrow \ldots \rightarrow A_n \rightarrow q$, $B = B_1 \rightarrow \ldots \rightarrow B_m \rightarrow q$
Then (a) is *not necessarily* equivalent to (b):

(a) $\Delta, A \ \vdash^? \ B_i, H * (q)$ succeeds
for $i = 1, \ldots, m$

(b) $\Delta, B \ \vdash^? \ A_i, H * (q)$ succeeds
for $i = 1, \ldots, n$.

To this regard, let

$$\Delta = \{a\}, \ A = a \rightarrow q, \ B = b \rightarrow q$$

Then $\Delta, A \ \vdash^? \ b, (q)$ is

$$c, a, a \rightarrow q \ \vdash^? \ b, (q)$$

while $\Delta, B \ \vdash^? \ a, (q)$ is

$$a, b \rightarrow q \ \vdash^? \ a, (q)$$

The first computation fails (we cannot restart here) and the second computation succeeds. It is easily seen that $\{c, a, a \rightarrow q, b \rightarrow q\}$ does not prove b, whereas it does prove a. We thus see that in the intuitionistic computation $\Delta \ \vdash^? \ q, H$, with bounded restart, backtracking is certainly necessary. The atom q can unify with any formula $A_1 \rightarrow \ldots \rightarrow A_n \rightarrow q$ and success or failure may depend on the choice of the formula. This gives an intuitive account of the difference of complexity between the intuitionistic and the classical case.

2.4 Conjunction and negation

The addition of conjunction to the propositional language does not change the system much. As we have seen at the beginning of the chapter, $\wedge$ can be fully characterized by the following two conditions on the consequence relation:

1. $\wedge$-Elimination rule
   $A \wedge B \vdash A$
   $A \wedge B \vdash B$

2. $\wedge$-Introduction rule
   $A, B \vdash A \wedge B$.

Let $\vdash$ be the smallest consequence relation for the language of $\{\to, \wedge\}$ closed under the deduction theorem and the $\wedge$ elimination and introduction rules.

We want to characterize $\vdash$ computationally. This we do using the following lemmas.

Lemma 2.4.1 *1.* $A \to B \to C \vdash A \wedge B \to C$

2. $A \wedge B \to C \vdash A \to B \to C$

3. $A \to B \wedge C \vdash (A \to B) \wedge (A \to C)$

4. $(A \to B) \wedge (A \to C) \vdash A \to (B \wedge C)$.

Proof. Exercise. □

Lemma 2.4.2 *Every formula A with conjunctions is equivalent in intuitionistic logic to a conjunction of formulae which contain no conjunctions.*

Proof. Use the equivalences of 2.4.1 to pull the conjunctions out. □

Definition 2.4.3 1. With any formula A with conjunctions we associate a unique (up to equivalence) set $C(A)$ of formulas as follows: given A, use 2.4.2 to present it as $\wedge A_i$, where each A_i contains no conjunctions. Let $C(A) = \{A_i\}$.

2. Let Δ be a set of formulae. Define $C(\Delta) = \bigcup_{A \in \Delta} C(A)$.

3. We can now define $\Delta \vdash^? A$, for Δ and A containing conjunctions. We will simply compute $C(\Delta) \vdash^? C(A)$, that is
$$C(\Delta) \vdash^? B \quad \forall B \in C(A).$$

4. The computation rule for conjunction can be stated directly: $\Delta \vdash^? A \wedge B$ succeeds iff $\Delta \vdash^? A$ succeeds and $\Delta \vdash^? B$ succeeds .

■

We now turn to **negation**. As we have seen negation can be introduced in classical and intuitionistic logic by adding a constant symbol $\perp$ for falsity and defining the new connective $\neg A$ for negation as $A \to \perp$.We will adopt this definition. However, we have to modify the computation rules, because we have to allow for the special nature of $\perp$, namely that $\perp \vdash A$ holds for any A.

Definition 2.4.4 [Computations for data and goal containing $\bot$ for intuitionistic and classical logic] The basic procedure is that one defined in 2.2.1, (plus the restart rule for classical logic), with the following modifications:

1. Modify (success) rule to read: $\Delta \vdash^? q$ immediately succeeds, if $q \in \Delta$ or $\bot \in \Delta$.
2. Modify (reduction rule) to read: from $\Delta \vdash^? q$ step, for $i = 1, \ldots, n$ to

   $$\Delta \vdash^? B_i$$

   if there is $C \in \Delta$ such that $Head(C) = q$ or $Head(C) = \bot$, and $Body(C) = (B_1, \ldots, B_n)$.

■

In 2.4.4 we have actually defined two procedures one is the computation without the restart rule for intuitionistic logic with $\bot$, and the other is the computation with the restart rule for classical logic. We have to show that indeed the two procedures correctly capture the intended fragment of the respective systems. This is easy to see. The rule $\bot \vdash A$ is built into the computation via the modifications in 1. and 2. of 2.4.4 and hence we know we are getting intuitionistic logic. To show that the restart rule yields classical logic, it is sufficient to show that the computation

$$(A \to \bot) \to \bot \vdash^? A$$

always succeeds with the restart rule. This can also be easily checked.

To complete the picture we show in the next proposition that the computation of $\Delta \vdash^? A$ with restart is the same as the computation of $\Delta \vdash^? (A \to \bot) \to A$ without restart. This means that the restart rule (with original goal A) can be effectively implemented by adding $A \to \bot$ to the database and using the formula $A \to \bot$ and the $\bot$-rules to replace uses of the restart rule. The above considerations correspond to the known translation from classical logic to intuitionistic logic, namely:

$\Delta \vdash A$ in classical logic iff $\Delta \vdash \neg A \to A$ in intuitionistic logic.

The proof is similar to that one of lemma 2.3.19, namely $Cop(G)$ is a way of representing $G \to \bot$ without using $\bot$ [3].

Proposition 2.4.5 *For any database Δ and goal G:*

$\Delta \vdash^? G$ succeeds with restart iff $\Delta \cup \{G \to \bot\} \vdash^? G$ succeeds without restart.

The above lemma can be used to prove that the restart rule can generalized to allow restart at any time, not only when we reach an atomic goal i.e.

from $\Delta \vdash^? A, (G_0)$ step to $\Delta \vdash^? G_0, (G_0)$

for any formula A. The soundness of this generalization may be easily derived from 2.4.5. Details are left to the reader.

[3]Recently, this result appears in [Nadathur 98] where a uniform proof system for classical logic is given.

Example 2.4.6 We check:

$$(a \to b) \to b \vdash^? (a \to \bot) \to b$$

We use rule for implication

$$(a \to b) \to b, a \to \bot \vdash^? b.$$

we have two choices here. One is to use the reduction rule with $a \to \bot$ and the second to use the rule for atoms with $(a \to b) \to b$. You can guess that the first case loops, so we use the second case:

$$(a \to b) \to b, a \to \bot \vdash^? a \to b$$

$$(a \to b) \to b, a \to \bot, a \vdash^? b$$

We use the reduction rule with $a \to \bot$

$$(a \to b) \to b, a \to \bot, a \vdash^? a$$

success.

■

Example 2.4.7 We check:

$$(q \to \bot) \to q \vdash^? q, (q)$$

rule for reduction

$$(q \to \bot) \to q \vdash^? q \to \bot, (q)$$

rule for implication

$$(q \to \bot) \to q, q \vdash^? \bot, (q)$$

Note that the rule for reduction involving $\bot$ can be used only when $\bot$ is in the database, i.e. in $q \to \bot \vdash^? a$ we can ask for q but *not* in the case of $q \to a \vdash^? \bot$. We cannot use the rule for reduction here. So, we fail in intuitionistic logic. In classical logic we can use restart to obtain:

$$(q \to \bot) \to q, q \vdash^? q, (q)$$

and terminate successfully. ■

Example 2.4.8

$$(((a \to \bot) \to b) \to \bot) \to c \vdash^? (a \to \bot) \to (b \to \bot) \to c$$

rule for implication

$$(((a \to \bot) \to b) \to \bot) \to c, a \to \bot \vdash^? (b \to \bot) \to c$$

rule for implication

$$(((a \to \bot) \to b) \to \bot) \to c, a \to \bot, b \to \bot \vdash^? c$$

The rule for reduction can be used with each of the three assumptions. If you want to use the system automatically try all three. Otherwise try the first one because it will make you *add* to the data and thus increase the chances of success.

$$((a \to \bot) \to b) \to \bot) \to c, a \to \bot, b \to \bot \vdash^? ((a \to \bot) \to b) \to \bot$$

rule for implication

$$(((a \to\perp) \to b) \to\perp) \to c, a \to\perp, b \to\perp, (a \to\perp) \to b \vdash^? \perp$$

rule for reduction with $b \to\perp$

$$(((a \to\perp) \to b) \to\perp) \to c, a \to\perp, b \to\perp, (a \to\perp) \to b \vdash^? \ b$$

rule for reduction with $(a \to\perp) \to b$

$$(((a \to\perp) \to b) \to\perp) \to c, a \to\perp, b \to\perp, (a \to\perp) \to b \vdash^? \ a \to\perp$$

rule for implication

$$(((a \to\perp) \to b) \to\perp) \to c, a \to\perp, b \to\perp, (a \to\perp) \to b, a \vdash^? \perp$$

rule for reduction using $a \to\perp$

$$(((a \to\perp) \to b) \to\perp) \to c, a \to\perp, b \to\perp), (a \to\perp) \to b, a \vdash^? \ a$$

success. ■

2.5 Disjunction

The handling of disjunction is much more difficult than the handling of conjunction and negation. Consider the formula $a \to (b \vee (c \to d))$. We cannot rewrite this formula in intuitionistic logic to anything of the form $B \to q$, where q is atomic (or $\perp$).

We therefore have to change our computation procedures to accommodate the general form of an intuitionistic formula with disjunction.

In classical logic disjunctions can be pulled out to the outside of formulas using the following rules:

1. $(A \vee B \to C) \equiv (A \to C) \wedge (B \to C)$
2. $(C \to A \vee B) \equiv (C \to A) \vee (C \to B)$.

Where $\equiv$ denotes logical equivalence (in our case logical equivalence in classical logic). (1) is valid in intuitionistic logic but (2) is not valid. In fact, if we add (2) as an axiom schema to intuitionistic logic we get a stronger logical system known as Dummett's **LC** (see chapter 4).

In intuitionistic logic we have the disjunction property, namely $\vdash_{\mathbf{I}} A \vee B$ iff $\vdash_{\mathbf{I}} A$ or $\vdash_{\mathbf{I}} B$. This is not true in classical logic **C**. Thus for example $\vdash A \vee (A \to B)$ but $\nvdash_{\mathbf{C}} A$ and $\nvdash_{\mathbf{C}} A \to B$. We have seen at the beginning of the chapter that the consequence relation rules defining disjunction are In view of the above we may want to adopt the computation rule below for disjunction in the goal.

R1: from $\Delta \vdash^? A \vee B$ step to $\Delta \vdash^? A$ or to $\Delta \vdash^? B$.

This corresponds to the consequence relation rules $A \vdash A \vee B$ and $B \vdash A \vee B$.

In case we have disjunction in the data, the rule is clear:

R2: from $\Delta, A \vee B \ \vdash^? \ C$ step to $\Delta, A \ \vdash^? \ C$ and to $\Delta, B \ \vdash^? \ C$.

This corresponds to the consequence relation rule

$$\frac{\Delta, A \vdash C \quad \Delta, B \vdash C}{\Delta, A \vee B \vdash C}$$

Let us adopt the above two rules for computation.

Example 2.5.1 $A \vee B \ \vdash^? \ A \vee B$. Using R1 we get

$$A \vee B \ \vdash^? \ A$$

or

$$A \vee B \ \vdash^? \ B$$

which fail. However using R2, we get

$A \ \vdash^? \ A \vee B$ and $B \ \vdash^? \ A \vee B$

which succeed using R1. ■

We can try to encorporate the two rules for disjunction within a goal-directed proof procedure for full intuitionistic logic. We tentatively propose the following definition.

Definition 2.5.2 Computation rules for full intuitionistic logic with disjunction.

1. The propositional language contains the connectives $\wedge, \vee, \rightarrow, \bot$. Formulae are defined inductively as usual.
2. We define the operation $\Delta + A$, for any formula $A = \bigwedge_i A_i$, as follows: $\Delta + A = \Delta \cup \{A_i\}$ provided A_i are not conjunctions.
3. The computation rules are as follows.

(suc) $\Delta \ \vdash^? \ q$ succeeds if $q \in \Delta$ or $\bot \in \Delta$;

(conj) from $\Delta \ \vdash^? \ A \wedge B$ step to $\Delta \ \vdash^? \ A$ and to $\Delta \ \vdash^? \ B$;

(g-dis) from $\Delta \ \vdash^? \ A \vee B$ step to $\Delta \ \vdash^? \ A$ or to $\Delta \ \vdash^? \ B$;

(imp) from $\Delta \ \vdash^? \ A \rightarrow B$ step to $\Delta + A \ \vdash^? \ B$;

(red) from $\Delta \ \vdash^? \ G$ if G is an atom q or $G = A \vee B$, for any $C \in \Delta$, with $C = A_1 \rightarrow \ldots A_n \rightarrow B$ step to

i. $\Delta \ \vdash^? \ A_i$, for $i = 1, \ldots, n$, and to

ii. $\Delta + B \ \vdash^? \ G$.

IT IS NOT CLEAR WHAT IS B IN THE ABOVE FORMULA: IS IT THE LAST CONSEQUENT OF C WHICH IS NOT AN IMPLICATION???

(c-dis) from $\Delta, A \vee B \ \vdash^? \ C$ step to $\Delta + A \ \vdash^? \ C$ succeeds andto $\Delta + B \ \vdash^? \ C$.

■

Notice that we must be allowed to perform a reduction step not only when the goal is atomic, but also when it is a disjunction, as in the following case

$$A, A \to B \vee C \vdash^? B \vee C$$

Similarly, even if the goal is an atom q we cannot require that B is atomic and $B = q$. Moreover, if there are disjunctions in the database, at every step we can choose to work on the goal or to split the database by (c-dis) rule. Moreover, if there are n formulas a systematic application of (c-dis) rule yields 2^n branches. All of this means that if we handle disjunction in the most obvious way we loose the goal-directedness of deduction and the computation results very inefficient. Can we do better? In the following we give an intuitionistic goal-directed procedure for data containing disjunction.

What makes difficult to define a goal-directed procedure in the intuitionistic case is that we must be prepared to switch the goal with another goal coming from another side of a disjunction, but at the same time we must keep the dependency between a goal and the database from which it is asked. In the proof procedure defined below we use labels to account for the dependency of a goal from the database it was asked and we use restart to handle disjunction. The labels in the database are partially ordered, that is they form a tree. This way of using labels is clearly reminiscent of the Kripke semantics of intuitionistic logic. The deduction-procedure defined below retains the goal-directness of the non-disjunctive case; in particular does not suffer of the two drawbacks of the nave-procedure we have seen above:

- we apply the reduction rule only when we meet an atomic goal;
- we split a disjunction $A \vee B$ in the database only if The "head" of A or B (to be defined properly) matches the current atomic goal.

At the end of the section, we will see how to obtain, as a particular case, a proof-procedure for classical logic with disjunctive formulas. However, in principle it is not even necessary as we can eliminate disjunction by translating $A \vee B$ as $(A \to B) \to B$, or $(A \to \bot) \wedge (B \to \bot) \to \bot$, although this translation may be not very efficient.

To simplify the proof procedure, we rewrite the formulas in a inessential way, by introducing the notion of D-formula, then we define a proof procedure for D-formulas.

Definition 2.5.3 A D-formulas is defined as

$$D = \ \bot \mid q \mid \bigwedge D \to \bigvee D$$

where $\bigwedge D$ and $\bigvee D$ are respectively finite conjunctions and disjunctions of D-formulas. ■

It is easy to see that every formula is equivalent to a set (or conjunction) of D-formulas.

When we consider a D-formula, we distinguish the following cases: either D is an atom, or it is $\bot$, or has the form $\bigwedge_{i=1}^m D_i \to \bigvee_{j=1}^n E_j$, where D_i and E_j are D-formulas and we assume that $n > 0$ and either $m > 0$ or $n > 1$. Thus, we can distinguish two subcases of non-atomic D-formulas:

(i) $D = \bigwedge_{i=1}^m D_i \to D'$ with $m > 0$ and D' is a D-formula, and
(ii) $D = \bigvee_{j=1}^n E_j$, with $n > 1$.

We need to define the Head of a D-formula in this context:

$Head(q) = \{q\}$,
$Head(\bigwedge_{i=1}^m D_i \to \bigvee_{j=1}^n E_j) = \bigcup_{j=1}^m Head(E_j)$.

Definition 2.5.4 A query has the form

$$\Gamma \ \vdash^? \ x : E, H$$

where E is a D-formula, Γ is a labelled set of D-formulas and constraints of the form $x \leq y$, (x, y are labels), H the history is a set of pairs of the form

$$\{(x_1, D_1), \ldots, (x_n, D_n)\},$$

where x_i are labels and D_i are D-formulas.

The proof rules for constraints are the following

- (≤ 1) $\Gamma \vdash x \leq x$;
- (≤ 2) $\Gamma \vdash x \leq y$, if $x \leq y \in \Gamma$;
- (≤ 3) $\Gamma \vdash x \leq y$, if for some z, $\Gamma \vdash x \leq z$ and $\Gamma \vdash z \leq y$.

The computation rule are the following.

- (success-atom) $\Gamma \ \vdash^? \ x : q, H$ immediately succeeds if $y : q \in \Gamma$ or $y : \bot \in \Gamma$ and $\Gamma \vdash y \leq x$.
- (implication)

  from $\Gamma \ \vdash^? \ x : \bigwedge_{i=1}^m D_i \rightarrow \bigvee_{j=1}^n E_j, H$

  step to

  $\Gamma, y : D_1, \ldots, y : D_m, x \leq y \ \vdash^? \ y : \bigvee_{j=1}^n E_j, H$, where $y \notin \Gamma$.
- (disjunction)

  from $\Gamma \ \vdash^? \ x : \bigvee_{j=1}^n E_j, H$ step to
  $\Gamma \ \vdash^? \ x : E_l, H \cup \{(x, \bigvee_{j \neq l}^n E_l)\}$ for some $l = 1, \ldots, n$
- (reduction) from $\Gamma \ \vdash^? \ x : q, H$, if there is a formula $y : D \in \Gamma$, such that $q \in Head(D)$ or $\bot \in Head(D)$, with $D = \bigwedge_{i=1}^m D_i \rightarrow \bigvee_{j=1}^n E_j$ and $\Gamma \vdash y \leq x$, then for some z such that $\Gamma \vdash y \leq z$ and $\Gamma \vdash z \leq x$ we step to

  $\Gamma \ \vdash^? \ z : D_i, H \cup \{(x : q)\}$ for $i = 1, \ldots, m$ and
  $\Gamma, z : E_j \ \vdash^? \ x : q, H$ for $j = 1, \ldots, n$.
- (restart) from

  $\Gamma \ \vdash^? \ x : q, H$, with q atomic

  if $(y, D') \in H$ step to

  $\Delta \ \vdash^? \ y : D', H \cup \{(x, q)\}$.
- (success-false) $\Gamma \ \vdash^? \ x : q, H$ immediately succeeds if $y : \bot \in \Gamma$.

■

Notice that the success-rule is a degenerate case of the reduction rule, (we have kept it distinct by imposing the constraints on the number of antecedents and consequents). Notice the presence of the success-false rule.

This is necessary in cases like the following (if we allow them):

$x : a, y : \bot, x \leq y \ \vdash^? \ x : c,$

here the success rule does not help. However, we do not need the success-false rule, if the starting database does not prove $x : \bot$ for any x. This is stated in lemma 2.5.8.

The proof system can be extended with a rule for conjunction.

from (conjunction) $\Gamma \ \vdash^? \ x : \bigwedge_{i=1}^{m} D_i$ step to
$\Gamma \ \vdash^? \ x : D_i$, for $i = 1, \ldots, m$.

Slight variants of the procedure are possible. For instance, we can make more goal-oriented the reduction rule. Rather than asking

$\Gamma, z : E_j \ \vdash^? \ x : q, H$ for $j = 1, \ldots, n,$

we can go on decomposing those E_j such that q or $\bot$ is in $Head(E_j)$. To this purpose we define a function which determines what are the next goals, or better the next queries in the case of reduction. This function $NEXT$ just applies the standard decomposition of sequent calculi, but it keeps the goal focused. The function $NEXT(\Delta, x : q, H, y : D)$, where D is a D-formula, $\Delta \ \vdash^? \ x : q, H$ is a query, and $\Delta \vdash y \leq x$, is defined as follows:

$NEXT(\Delta, x : q, H, y : D) = \Delta \ \vdash^? \ x : q, H$ if $\{q, \bot\} \cap Head(D) = \emptyset$
$NEXT(\Delta, x : q, H, y : D) = \{\Delta \ \vdash^? \ z : D_i, H \cup \{(x, q)\} \mid i = 1, \ldots, m\} \cup \bigcup_j NEXT(\Delta \cup \{z : E_j\}, x : q, H, z : E_j),$
for one z such that $\Gamma \vdash y \leq z$ and $\Gamma \vdash z \leq x$ if $D = \bigwedge_{i=1}^{m} D_i \rightarrow \bigvee_{j=1}^{n} E_j$.

Then the reduction rule becomes

from $\Gamma \ \vdash^? \ x : q, H$, if there is a formula $y : D \in \Gamma$, such that $q \in Head(D)$ or $\bot \in Head(D)$, and $\Gamma \vdash y \leq x$, step to every $Q \in NEXT(\Gamma, x : q, H, y : D)$.

It can be proved that the reduction rule using NEXT is equivalent to the more general reduction rule.

Example 2.5.5 Let $\Delta = \{x : A_1, \ldots, x : A_5\}$ where:

$$\begin{aligned} A_1 &= \ a \rightarrow (b \rightarrow c) \vee (d \rightarrow e), \\ A_2 &= \ c \rightarrow p, \\ A_3 &= \ (b \rightarrow p) \rightarrow t, \\ A_4 &= \ q \rightarrow d, \\ A_5 &= \ (q \rightarrow e) \rightarrow s. \end{aligned}$$

In figure 2.3 it is shown a derivation of

$\Delta \ \vdash^? \ x : a \rightarrow t \vee s, \emptyset$

which correspond to check $A_1 \wedge \ldots \wedge A_5 \vdash a \rightarrow (t \vee s)$ in intuitionistic logic. We adopt the follwing abbreviations: we omit Δ and in each node we only show the additional data whenever is introduced in the database (thus, the complete database in each query is given by Δ plus the formulas in each database on the branch from the root to that query). Moreover we omit the history as it is clear from the structure of the tree, we make an exception for $(y : t)$ which is used in the (unique) restart step. If we use the formulation with the function NEXT, from (*) we immediately step to (i),(ii), and (iii) without any intermediate step.

■

When we apply the reduction rule as $\Gamma \vdash^? x : q, H$ with respect to a formula $y : D \in \Gamma$, $D = \bigwedge_{i=1}^{m} D_i \to \bigvee_{j=1}^{n} E_j$, we have to choose a label z such that $\Gamma \vdash y \leq z$ and $\Gamma \vdash z \leq x$ and we must step to

(1) $\Gamma \vdash^? z : D_i, H \cup \{(x : q)\}$ for $i = 1, \ldots, m$ and to
(2) $\Gamma, z : E_j \vdash^? x : q, H$ for $j = 1, \ldots, n$.

In general we cannot fix this z in advance. It must be a minimal z such that (1) succeeds. In specific cases we can determine z in advance: if $n = 1$ and E_n is atomic, we can always take $z = x$. If $m = 0$, we can always take $z = y$. The correctness of these choice follows from the the property of monotonicity stated in lemma 2.5.8.

In order to prove the soundness of the procedure, we need to introduce the notion of realization of a database.

Definition 2.5.6 Given Δ as above, let $M = (W, \leq_M, a_0, V)$ be a Kripke model. A *realization* of Γ is a mapping $\rho : \mathcal{A} \to W$, such that

- $x \leq y \in \Gamma$ implies $\rho(x) \leq \rho(y)$;
- $x : D \in \Gamma$ implies $M, \rho(x) \models D$.

Given a query $Q = \Gamma \vdash x : E, H$, we say that Q is valid iff for all M and all realization ρ of Γ in M, we have:

either $M, \rho(x) \models E$, or for some $(y, F) \in H$, $M, \rho(y) \models F$.

■

Theorem 2.5.7 (Soundness) *If $\Delta \vdash x : E, H$ is derivable then is valid.*

Proof. By induction on the length of computations. Details are left to the reader. □

In order to prove completeness we need to show that the cut-rule, suitably formulated, is admissible. By $\Gamma[x : D]$, we denote that $x : D \in \Gamma$. The proof makes use of some (almost) obvious properties of the deduction procedure which are stated in the following lemma.

Lemma 2.5.8 • *(i) (Monotony) if $\Gamma \vdash^? x : D, H$ succeeds and $\Gamma \subseteq \Delta$, $\Delta \vdash x \leq y$, $H \subseteq H'$, then also $\Delta \vdash^? y : D, H'$ succeeds.*

- *(ii) $\Gamma \vdash^? x : D, H \cup \{(y : E)\}$ succeeds iff $\Gamma \vdash^? y : E, H \cup \{(x : D)\}$ succeeds.*
- *(iii) if $\Gamma \vdash^? x : D, H \cup \{(x : D)\}$ succeeds then also $\Gamma \vdash^? x : D, H$ succeeds.*
- *(iv) if $\Gamma \vdash^? x : D, H$ succeeds and for no label y $\Gamma \vdash^? y : \bot, H$ succeeds, then $\Gamma \vdash^? x : D, H$ succeeds without using the success-false rule.*

- *(v) if $\Gamma \vdash^? x : \bot, H$ succeeds, then for any D $\Gamma \vdash^? y : D, H$ succeeds.*

Theorem 2.5.9 *If $\Gamma[x : D] \vdash^? y : D_1, H_1$ and $\Delta \vdash^? z : D, H_2$ succeed,then also*

$$\Gamma[x : D/\Delta, z] \vdash^? y[x/z] : D_1, H_1[x/z] \cup H_2 \text{ succeed,}$$

where $\Gamma[x : D/\Delta, z] = (\Gamma - \{x : D\})[x/z] \cup \Delta$.

Proof. The proof of this theorem is similar to the one of theorem 2.2.5 and proceeds by induction on pairs (c, h), where c is the complexity of D and h is the height of a derivation of a successful derivation of $\Gamma[x : D] \vdash^? y : D_1, H_1$. The precise definition of complexity does not matter, we only need that the atoms have a minimal complexity and that given D of the form $\bigwedge_{i=1}^k D_i \rightarrow \bigvee_{j=1}^n E_j$, the complexity of D_i and of E_j is smaller than that of D. We only consider the most difficult case, when $\Gamma[x : D] \vdash^? y : D_1, H_1$ succeeds by recution with respect to D. In this case D_1 is an atom, say q and either $q \in Head(D)$ or $\bot \in Head(D)$, $\Gamma \vdash x \leq y$, and for some u such that $\Gamma \vdash x \leq u$ and $\Gamma \vdash u \leq y$ we step to

$\Gamma \vdash^? u : D_i, H_1 \cup \{(y : q)\}$ for $i = 1, \ldots, k$ and
$\Gamma, u : E_j \vdash^? y : q, H_1$ for $j = 1, \ldots, n$.

which succeed with a smaller height. By induction hypothesis, we get that

(a_i) $\Gamma[x : D/\Delta, z] \vdash^? u[x/z] : D_i, (H_1 \cup \{(y : q)\})[x/z] \cup H_2$ for $i = 1, \ldots, k$ and
(b_j) $(\Gamma \cup \{u : E_j\})[x : D/\Delta, z] \vdash^? y[x/z] : q, H_1[x/z] \cup H_2$ for $j = 1, \ldots, n$.

By the second premise, we get that for some $v \notin \Delta$, we have that

(c) $\Delta, z \leq v, v : D_1, \ldots, v : D_k \vdash^? v : \bigvee_{j=1}^n E_j, H_2$ succeeds.

Since each D_i has a smaller complexity than D, by induction hypothesis we can repeatedly cut (a_i) and (c), so that we get that

$$\Gamma[x : D/\Delta, z] \vdash^? u[x/z] : \bigvee_{j=1}^n E_j, (H_1 \cup \{(y : q)\})[x/z] \cup H_2$$

succeeds. But this implies that also

(d_1) $\Gamma[x : D/\Delta, z] \vdash^? u[x/z] : E_1, H_1[x/z] \cup \{(y[x/z] : q), (u[x/z], \bigvee_{j=2}^n E_j)\} \cup H_2$ succeeds.

Since E_1 has a smaller complexity than D, again by induction hypothesis we can cut (d_1) and (b_1) so that we get

$$\Gamma[x : D/\Delta, z] \vdash^? y[x/z] : q, H_1[x/z] \cup \{(y[x/z] : q), (u[x/z], \bigvee_{j=2}^n E_j)\} \cup H_2 \text{ succeeds.}$$

By the previous lemma, we get that also $\Gamma[x : D/\Delta, z] \vdash^? u[x/z] : \bigvee_{j=2}^n E_j H_1[x/z] \cup \{(y[x/z] : q), \} \cup H_2$ succeeds, whence

(d_2) $\Gamma[x : D/\Delta, z] \vdash^? u[x/z] : E_2, H_1[x/z] \cup \{(y[x/z] : q), (u[x/z], \bigvee_{j=3}^n E_j)\} \cup H_2$ succeeds

By induction hypothesis, we can cut (d_2) and (b_2). By repeating this argument up to n we finally get that

$$\Gamma[x : D/\Delta, z] \vdash^? y[x/z] : q, H_1[x/z] \cup \{(y[x/z] : q)\} \cup H_2 \text{ succeed,}$$

so that by the previous lemma the claim follows. □

Theorem 2.5.10 *If* $\Gamma \ \vdash^? \ x : D, H$ *is valid then it succeeds.*

Proof. We prove the contrapositive, i.e. if $\Gamma \ \vdash^? \ x : D, H$ does not succeed then it is not valid. We construct a Kripke model by extending the database, through the evaluation of all possible formulas at every world (each represented by one label) of the database. Since such evaluation may lead, for implication formulas, to create new worlds, we must carry on the evaluation process on these new worlds. For this reason we consider in the construction an enumeration of pairs (x_i, D_i), where x_i is a label and D_i is a D-formula.

Assume $\Gamma \ \vdash^? \ x : D, H$ fail. We let $\mathcal{A}$ be a denumerable alphabet of labels and $\mathcal{L}$ be the underlying propositional language. Let (x_i, D_i), for $i \in \omega$ be an enumeration of pairs of $\mathcal{A} \times \mathcal{L}$, starting with the pair (x, D) and containing infinitely many repetitions, that is

$$(x_0, D_0) = (x, D),$$
$$\forall y \in \mathcal{A}, \forall E \in \mathcal{L}, \forall n \ \exists m > n \ (y, E) = (x_m, D_m).$$

Given such enumeration we define i) a sequence of databases Γ_n, ii) a sequence of histories H_n, iii) a new enumeration of pairs (y_n, E_n), as explained below. The construction, will depend at each stage on the form of the formula E_n under examination. We refer to the notation for D-formulas introduced at the beginning of the section, i.e. either D is an atom, or it is $\bot$, or $D = \bigwedge_{i=1}^m F_i \rightarrow D'$ with $m > 0$, or $D = \bigvee_{j=1}^n F_j$, with $n > 1$, where F_i, G_j, D' are D-formulas.

- (step 0) Let $\Gamma_0 = \Gamma$, $H_0 = H$, $(y_0, E_0) = (x, D)$.
- (step n+1) Given (y_n, E_n), we let

  - $\Gamma_{n+1} = \Gamma_n$,
  - $(y_{n+1}, E_{n+1}) = (x_{k+1}, D_{k+1})$, where $k = max_{t \leq n} \exists s_{\leq n} (y_s, E_s) = (x_t, D_t)$,
  - $H_{n+1} = H_n$

  The above definition is modified as shown when $y_n \in Lab(\Gamma_n)$ and we are in one of the following cases:

  – $E_n = q$ (q is an atom) or $E_n = \bigvee_j^m G_j$, and $\Gamma_n \ \vdash^? \ y_n : E_n, H_n$ fails, in this case we let $H_{n+1} = H_n \cup \{(y_n, E_n)\}$.

  – $E_n = \bigwedge_{j=1}^m F_j \rightarrow D'$ and $\Gamma_n \ \vdash^? \ y_n : E_n, H_n$ fails, we let

  $\Gamma_{n+1} = \Gamma_n \cup \{y_n \leq x_s, x_s : F_1, \ldots, x_s : F_m\}$,
  $(y_{n+1}, E_{n+1}) = (x_m, D')$,

  where $x_s = min\{x_t \in \mathcal{A} \mid x_t \notin Lab(\Gamma_n)\}$.

  – $E_n = \bigvee_j^m G_j$, and $\Gamma_n \ \vdash^? \ y_n : E_n, H_n$ succeeds, but

  for $j = 1, \ldots, m$, $\Gamma_n \ \vdash^? \ y_n : \bigvee_{j \neq l}^m G_j, H_n$ fails.

  we let $H_{n+1} = H_n \cup \{(y_n, \bigvee_{j \neq l}^m G_j)\}$ for one $1 \leq l \leq m$.

Each stage of the construction defines, so to say, a new query. Intuitively, we follow the enumeration (x_n, D_n) in order to determine what is the formula and world to consider at the next stage, unless the formula currently evaluated is a conditional (and it fails). In such a case, we evaluate its consequent at a new-created world. When we come to an atomic formula, or to a disjunction. we go back to the enumeration (x_n, D_n) to pick the next pair. The proof of the theorem is composed be the next lemmas. □

Lemma 2.5.11 $\forall k\ \exists n \geq k\ (x_k, D_k) = (y_n, E_n)$.

Proof. By induction on k. If $k = 0$, the claim hold by definition. Let $(x_k, D_k) = (y_n = E_n)$.

- (i) if either $y_n \notin Lab(\Gamma_n)$, or $\Gamma_n, \alpha_n \ \vdash^? \ y_n : E_n, H_n$ succeeds, or E_n is atomic, or it is a disjunction, then $(x_{k+1}, D_{k+1}) = (y_{n+1}, E_{n+1})$.
- (ii) Otherwise, let $E_n = \bigwedge_{j=1}^m F_j \to G_1 \to \ldots \to G_t \to K$, where K is not an implication ($t \geq 0$), then

  $$(x_{k+1}, D_{k+1}) = (y_{n+t+1}, E_{n+t+1}).$$

□

Lemma 2.5.12 *In the hypothesis* $\Gamma_0 \ \vdash^? \ x_0 : D_0, H_0$ *fails, the following holds: for all* $n \geq 0$, *if* $(y, E) \in H_n$, *or* $E = \bot$, *then* $\Gamma_n \ \vdash^? \ y : E, H_n$ *fails.*

Proof. By lemma 2.5.8 we consider only the case of $(y : E) \in H$. We proceed by induction on n. Suppose it does not hold, let n be the minimum stage for which it does not hold. By hypothesis and lemma 2.5.8, we can assume $n > 0$. Let $n = m + 1$ and suppose the property holds up to m. We need only to consider the cases when $\Gamma_m \neq \Gamma_{m+1}$ or $H_m \neq H_{m+1}$.

- Let E_m be an atom or a disjunction, and $H_{m+1} = H_m \cup \{(y_m, E_m)\}$. Then $\Gamma_m \ \vdash^? \ y_m : E_m, H_m$ fails. Suppose for some $(y, E) \in H_{m+1}$, $\Gamma_m \ \vdash^? \ y : E, H_{m+1}$ succeeds. By lemma 2.5.8(ii) it must be $(y : E) \neq (y_m, E_m)$, thus $(y, E) \in H_m$, then by hypothesis and lemma 2.5.8(iii), we get a contradiction.
- Let $E_m = \bigwedge_{j=1}^k F_j \to D'$ and $\Gamma_m \ \vdash^? \ y_m : E_m, H_m$ fails, then

  $$\Gamma_{m+1} = \Gamma_m \cup \{y_m \leq x_t, x_t : F_1, \ldots, F_k\},$$
  $$(y_{m+1}, E_{m+1}) = (x_t, D'),$$

  where $x_t = min\{x_s \in \mathcal{A} \mid x_s \notin Lab(\Gamma_m)\}$. Suppose for some $(y, E) \in H_{m+1} = H_m$,

  (*) $\Gamma_{m+1} \ \vdash^? \ y : E, H_{m+1}$ succeeds,

  consider the following computation of $\Gamma_m \ \vdash^? \ y_m : E_m, H_m$. Start with

  $$\Gamma_m \ \vdash^? \ y_m \bigwedge_{j=1}^k F_j \to D', H_m,$$

  step by the implication rule to

  $$\Gamma_{m+1} \ \vdash^? \ y_{m+1} : D', H_{m+1},$$

  go on with the computation until we reach an atomic goal, let us say $\Sigma \ \vdash^? \ z : q, H'$. Since $H_{m+1} \subseteq H'$, we can step by restart to

  $$\Sigma \ \vdash^? \ y : E, H' \cup \{(z : q)\}.$$

  Since $\Gamma_{m+1} \subseteq \Sigma$, by (*) and monotony, the above query succeeds, whence $\Gamma_m \ \vdash^? \ y_m : E_m, H_m$ succeeds, contradicting the hypothesis.

- $E_m = \bigvee_j^t G_j$, and $\Gamma_m \vdash^? y_m : E_m, H_m$ succeeds, but for $j = 1, \ldots, t$, $\Gamma_m \vdash^? y_m : \bigvee_{j\neq l}^t G_j, H_m$ fails, then $H_{m+1} = H_m \cup \{(y_m, \bigvee_{j\neq l_0}^t G_j)\}$ for one $1 \leq l_0 \leq t$. Suppose for some $(y : E) \in H_{m+1}$ $\Gamma_{m+1} \vdash^? y : E, H_{m+1}$ succeeds, by lemma 2.5.8(ii), we get

  $\Gamma_{m+1} \vdash^? y_m : \bigvee_{j\neq l_0}^t G_j H_{m+1}$ succeeds,

  so that by lemma 2.5.8(iii), since $\Gamma_{m+1} = \Gamma_m$, we get that $\Gamma_m \vdash^? y_m : \bigvee_{j\neq l_0}^t G_j, H_m$ succeeds, against the hypothesis.

□

Lemma 2.5.13 *For all $n \geq 0$, if $\Gamma_n \vdash^? y_n : E_n, H_n$ fails, then:*

$\forall m \geq n$ $\Gamma_m \vdash^? y_n : E_n, H_m$ fails.

Proof. By induction on the complexity of E_n. If $E_n = \bot$ or $E_n = q$, or E_n is a disjunction, the claim immediately follows by construction and the previous lemma, without actually using the induction hypothesis. Let $E_n = \bigwedge_{j=1}^m F_j \to D'$ and $\Gamma_n \vdash^? y_n : E_n, H_n$ fails, then

$$\Gamma_{n+1} = \Gamma_n \cup \{y_n \leq x_t, x_t : F_1, \ldots, x_t : F_k\}, ,$$

where $x_t = min\{x_s \in \mathcal{A} \mid x_s \notin Lab(\Gamma_n)\}$. By the computation rules, $\Gamma_{n+1} \vdash^? y_t : D', H_{n+1}$ fails; since $(y_{n+1}, E_{n+1}) = (x_t, D')$, by induction hypothesis, we have that

(*) for all $m \geq n+1$, $\Gamma_m \vdash^? y_{n+1} : D', H_m$ fails.

Suppose for some $m > n$, $\Gamma_n \vdash^? y_n : \bigwedge_{j=1}^k F_j \to D', H_n$ succeeds, then for some label $u \notin Lab(\Gamma_m)$,

(1) $\Gamma_m \cup \{y_n \leq u, u : F_1, \ldots, u : F_k\} \vdash^? u : D', H_m$ succeeds.

On the other hand by monotony, being $\Gamma_{n+1} \subseteq \Gamma_m$, we easily get that

(2) $\Gamma_m \vdash^? y_{n+1} : F_j, H_m$ succeeds for $j = 1, \ldots, t$.

Since $y_n \leq y_{n+1} \in \Gamma_m$, By cutting (1) and (2), we get that $\Gamma_m \vdash^? y_{n+1} : D', H_m$ succeeds, against (*).

□

Lemma 2.5.14 *$\forall m, \Gamma_m \vdash^? x : D, H_m$ fails.*

Proof. Immediate by the previous lemma. □

Lemma 2.5.15 *If $E_n = \bigwedge_{j=1}^t F_j \to D'$ and*

$\Gamma_n \vdash^? y_n : \bigwedge_{j=1}^t F_j \to D'$,H_n fails,

then there is a $y \in \mathcal{A}$, such that for $k \leq n$, $y \notin Lab(\Gamma_k)$ and $\forall m > n$:

(i) $y_n \leq y \in \Gamma_m$,
(ii) $\Gamma_m \vdash^? y : F_j, H_m$ succeeds for $j = 1, \ldots, t$,
(iii) $\Gamma_m \vdash^? y : D', H_m$ fails.

Proof. By construction, we can take $y = y_{n+1}$, the new point created at step $n+1$, so that (i),(ii),(iii) hold for $m = n+1$. In particular,

(*) $\Gamma_{n+1} \vdash^? y_{n+1} : D, H_{n+1}$ fails.

Since the Γ_m are not decreasing by monotonicity, we immediately have that (i) and (ii) also hold for every $m > n+1$. By construction, we know that $E_{n+1} = D'$, whence by (*) and lemma 2.5.13, (iii) also holds for every $m > n+1$. □

Lemma 2.5.16 *If for some n $\Gamma_n \vdash^? y : \bigvee_j^t G_j, H_n$ succeeds, then there is an $m > n$, such that for some G_l, with $1 \leq l \leq t$, $\Gamma_m \vdash^? y : G_l, H_m$ succeeds.*

Proof. Let $\Gamma_n \vdash^? y : \bigvee_j^t G_j, H_n$ succeeds, and let $k \geq n$, such that $(y, \bigvee_j^t G_j)$ is considered at step k, that is $(y, \bigvee_j^t G_j) = (y_k, E_k)$, then $\Gamma_k \vdash^? y_k : E_k, H_k$ succeds. If (a) for some G_l, $\Gamma_k \vdash^? y_k : G_j, H_k$ succeeds, we are done. On the other hand if (b) for every l, $\Gamma_k \vdash^? y_k : \bigvee_{j \neq l} G_j, H_k$ fails, then it is easy to see $\Gamma_{k+1} \vdash^? y_k : G_l, H_{k+1}$ for some l succeeds, and we are done again. Otherwise, $\Gamma_k \vdash^? y_k : \bigvee_{j \neq l} G_j, H_k$ succeeds for some l. Then, as before, there is $h \geq k$, such that $(y_k, \bigvee_{j \neq l} G_j) = (y_h, E_h)$ is considered at step h, we can repeat the argument (at most $t-2$ times) until we fall in case (a) or in case (b). □

Construction of the Canonical model
We define an intuitionistic Kripke-model as follows $M = (W, u_0, \leq, V)$, such that

- $W = \bigcup_n Lab(\Gamma_n) \cup \{u_0\}$, where $u_0 \notin \bigcup_n \Gamma_n$;
- $x \leq y \equiv x = u_0 \vee \exists n\ \Gamma_n \vdash x \leq y$,
- $V(u_0) = \emptyset$,
-$V(x) = \{q \mid \exists n\ x \in Lab(\Gamma_n)\ \wedge\ \Gamma_n \vdash^? x : q, H_n \text{ succeeds}\}$ for $x \neq u_0$.

It is easy to see that $\leq$ is reflexive and transitive and that V is monotonic with respect to $\leq$.

Lemma 2.5.17 *for all $x \in W, x \neq u_0$ and D-formulas E,*

$$M, x \models E \ \Leftrightarrow\ \exists n\ x \in Lab(\Gamma_n) \wedge \Gamma_n \vdash^? x : E, H_n \textit{ succeeds.}$$

Proof. We prove both directions by mutual induction on $cp(E)$. If E is an atom then the claim holds by definition, if E is $\bot$ it follows by lemma 2.5.12. Assume $E = \bigwedge_{j=1}^t F_j \to D'$.
($\Leftarrow$) Suppose for some m $\Gamma_m, \alpha_m \vdash^? x : \bigwedge_{j=1}^t F_j \to D', H_m$ succeeds. Let $x \leq y$ and $M, y \models \bigwedge_{j=1}^t F_j$, for some y. Then, $M, y \models F_j$ for $j = 1, \ldots, t$. By definition of $\leq$ we have that for some n_1, $\Gamma_{n_1} \vdash x \leq y$ holds. Moreover, by induction hypothesis, for some m_j, $j = 1, \ldots, t$, $\Gamma_{n_j} \vdash^? y : F_j, H_{m_j}$ succeeds. Let $k = max\{m_1, \ldots, m_t, n_1, m\}$, then we have

(1) $\Gamma_k \vdash^? x : \bigwedge_{j=1}^t F_j \to D', H_k$ succeeds,
(2) $\Gamma_k \vdash^? y : F_j, H_k$ succeeds for all $j = 1, \ldots, t$,
(3) $\Gamma_k \vdash x \leq y$.

So that from (1) we also have:

(1') $\Gamma_k \cup \{x \leq z, z : F_1, \ldots, z : F_t\} \vdash^? z : D', H_k$ succeeds, (with $z \notin Lab(\Gamma_k)$).

We can cut (1') and (2), and obtain that:

$\Gamma_k \vdash^? y : D', H_k$ succeeds.

and by induction hypothesis, $M, y \models D$.

($\Rightarrow$) Suppose by way of contradiction that $M, x \models \bigwedge_{j=1}^{t} F_j \to D'$, but for all n, if $x \in Lab(\Gamma_n)$, then $\Gamma_n \ \vdash^? \ x : \bigwedge_{j=1}^{t} F_j \to D', H_n$ fails. Let $x \in Lab(\Gamma_n)$, then there are $m \geq k > n$, such that $(x, \bigwedge_{j=1}^{t} F_j \to D') = (y_m, E_m)$ is considered at step $m+1$, so that we have:

$\Gamma_m \ \vdash^? \ y_m : \bigwedge_{j=1}^{t} F_j \to D', H_m$ fails.

By lemma 2.5.15, there is a $y \in \mathcal{A}$, such that (a) for $t \leq m$, $y \notin Lab(\Gamma_t)$ and (b):

$\forall m' > m$
(i) $\Gamma_{m'} \vdash y_m \leq y$,
(ii) $\Gamma_{m'} \ \vdash^? \ y : F_j, H_{m'}$ succeeds for all $j = 1, \ldots, t$
(iii) $\Gamma_{m'} \ \vdash^? \ y : D', H_{m'}$ fails.

By (b)(i) we have $x \leq y$ holds, by (b)(ii) and induction hypothesis, we have $M, y \models \bigwedge_{j=1}^{t} F_j$. By (a) and (b)(iii), we get that

$\forall n$ if $y \in Lab(\Gamma_n)$, then $\Gamma_n, \alpha_n \ \vdash^? \ y : D', H_n$ fails.

Hence, by induction hypothesis, we have $M, y \not\models D'$, and we get a contradiction.

Let $E = \bigvee_{j=1}^{t} G_j$. ($\Leftarrow$) Let $\Gamma_n \ \vdash^? \ x : \bigvee_{j=1}^{t} G_j, H_n$ succeeds, then by lemma 2.5.16, there is some $m \geq n$ such that $\Gamma_m \ \vdash^? \ x : G_l, H_m$ succeeds for $l = 1, \ldots, t$. By induction hypothesis, we have $M, x \models G_l$. ($\Rightarrow$) If $M, x \models E$, then for some j, $M \models G_j$, then we simply apply the induction hypothesis and conclude. □

Proof of Completeness Theorem 2.2.6. We are able now to conclude the proof of the completeness theorem. Let $\rho(z) = z$, for every $z \in Lab(\Gamma)$, where Γ is the original database. It is easy to see that ρ is a realization of Γ_0 in M and if $u : C \in \Gamma$, then by identity and the previous lemma we have $M, \rho(u) \models C$. On the other hand, by lemmas 2.5.14,2.5.12, 2.5.17 we have $M, \rho(x) \not\models D$ and for all $(y, E) \in H$, $M, \rho(y) \not\models E$. This concludes the proof. ■

We end the section with a remark on the treatment of disjunction in classical logic. To have classical disjunction we can adopt the procedure for intuitionistic logic and ignore the constraints on the labels (whence the labels themselves). In case of classical logic, we can think that there is a single world, so that for any pair of labels x, y, the constraint $x \leq y$ is satisfied.

Example 2.5.18 Let us check $a \vee (b \to c) \vdash (b \to a) \vee c$ in classical logic. The derivation is shown in Figure 2.4. Let $\Delta = \{x : a \vee (b \to c)\}$. Query $(*)$ fails in intuitionistic logic as $y \not\leq x$, whereas it succeeds in classical logic since b is in the database, regardless of the label. ■

The procedure for classical logic can be further simplified in many ways according to the syntax of the formulas we want to treat (or the amount of rewriting we are willing to perform). One radical reduction is the following: any formula can be classically transformed into a set of *clauses* C of the form

$C = p_1 \wedge \ldots p_n \to q_1 \vee \ldots \vee q_m$, where $m > 0$ or $m = 1$ and $q_1 = \bot$

We assume that goals are just atoms. This restricted pattern of clauses and goals is however sufficient to encode any tautology problem in classical logic. We can even further eliminate $\bot$ introducing new atoms, although we will not do it. For database and goals of this format, the restart rule can be restricted to the initial goal (the proof is similar to the one of proposition 2.3.17). Thus, we may write a query as $\Delta \ \vdash^? \ q, (q_0)$. For database and goal of the above form the rules of definition 2.5.4 simplify to the following:

- (success) $\Delta \vdash^? q, (q_0)$ immediately succeeds if $q \in \Delta$;
- (reduction) from $\Delta \vdash^? q, (q_0)$, if there is a clause $p_1 \wedge \ldots \wedge p_n \rightarrow q_1 \vee \ldots \vee q_m \in \Delta$ such that $q = q_i$ or $m = 1$ and $q_1 = \bot$ step to

  (1) $\Delta \vdash^? p_j, (q_0)$ for $j = 1, \ldots, n$ and to (2) $\Delta, q_j \vdash^? q, (q_0)$ for $j = 1, \ldots, n \wedge j \neq i$.

  If $m = 1$ and $q_1 = \bot$ step only to (1).
- (restart) from $\Delta \vdash^? q, (q_0)$ step to $\Delta \vdash^? q_0, (q_0)$.

This rules are almost identical to the propositional version of the rules given recently by Nadathur [Nadathur 98] (for first-order classical logic).

2.6 The $\forall, \rightarrow$-fragment of intuitionistic logic

In this last section we present a goal-directed procedure for the $\forall, \rightarrow$ fragment of intuitionistic logic. This procedure is in the style of a logic programming proof-procedure which use unification and computes answer-substitutions, that is the outcome of successful computation of

$$\Delta \vdash^? G[x],$$

where $G[x]$ stands for $\exists x G[x]$ is not only 'yes', but is a most general substitution x/t, such that $\Delta \vdash G[x/t]$ holds in intuitionistic logic. In the literature [Miller et al. 91],[Miller 92] similar extensions have been proposed, but with the exception of [Gabbay and Reyle 93] and [Gabbay 92] the answer computation mechanism is never discussed in detail. We consider it inherent to the extension of the logic programming paradigm, rather than part of the implementation details which come as an afterthought.

Although a broader extension of our methodology to first-order languages, is out of the scope of the present work, yet what we present in this section may give some hint on how to extend the goal-directed methods to the first-order language for other logics. We assume known standard notions and notations relative to first-order languages, (we refer the reader to any standard textbook as [Gallier 87]). We will however reformulate the Kripke semantics of intuitionistic logic for the first-order case.

In classical logic one can always put formulas in prenex form, replace existential quantifiers by Skolem functions, and use the unification mechanism to deal with so-obtained universal sentences. In intuitionistic logic, as well as in modal logics one cannot skolemize at the beginning of computation. The process of eliminating existential quantifiers by Skolem functions must be carried on in parallel with goal reduction. This process has been called "run-time skolemization" in [Gabbay 92, Gabbay and Reyle 93] and adopted in N-Prolog [Gabbay and Reyle 93] [4].

The consequence-relation rules for the universal quantification are the following:

($\forall$-I) :

$$\frac{\Gamma \vdash A[c]}{\Gamma \vdash \forall x A[x],}$$

provided c is a new constant which does not occur neither in Δ nor in A.

($\forall$-E) $\forall x A[x] \vdash A[x/t]$, where t is any term.

[4] A similar idea for classical logic is embodied in Free-Variable tableaux originally proposed by Fitting for classical first order logic [] IS IT RIGHT THIS CONNECTION????.

ARE THESE TWO RULES (ADDED TO THE CONSEQUENCE RULES FOR THE IMPLICATIONAL FRAGMENT) SUFFICIENT TO CHARACTERIZE THE $\forall, \rightarrow$ FRAGMENT OF INTUITIONISTIC LOGIC???

When used backwards the ($\forall$-I) rule says that, in order to prove $\forall x A[x]$, one has to prove $A[c]$ for a new constant c. To incorporate this rule soundly within the goal-directed proof procedure for intuitionistic logic requires some care.

Example 2.6.1 The formula

$$\forall x((p(x) \rightarrow \forall y p(y)) \rightarrow q) \rightarrow q$$

is not a theorem of intuitionistic logic, but we if we apply the rule ($\forall$-I) naively we succeed as shown in Fig 2.5. ■

In the shown derivation, c is a new constant, we succeed since we can unify $p(x)$ and $p(c)$. We should block the unification of x with c. The rule should prevent the unification of a free variable with a constant which is introduced *later*. This might be done by supplying information on when a constant has been introduced (a sort of "time-stamping" of constants).

We will follow the alternative approach of *run-time skolemization*. The idea is to eliminate the universal quantifiers by introducing a new Skolem function $c(x)$ which depends on all free-variables occurring in the database and in the goal [5]. In the example above, we block unification, since $c(x)$ and x cannot unify by the occur-check.

Before we define the proof procedure, we fix some notation. We consider formulas of a first-order language containing the logical constants $\forall, \rightarrow$, and constant, function and predicate symbols of each arity. We assume known the notion of term. The notation $Var(t)$ denotes the set of variables occurring in a term t.

Definition 2.6.2 We simultaneously define formulas, and the set of bounded ($BVAR$) and free variables ($FVAR$) occurring in a formula.

- If R is a n-ary predicate symbol and $t_1, \ldots t_n$ is a tuple of terms, then $R(t_1, \ldots, t_n)$ is a formula; we call $R(t_1, \ldots, t_n)$ an atom. We let $BVAR(R(t_1, \ldots, t_n)) = \emptyset$ and $Fvar(R(t_1, \ldots, t_n)) = Var(t_1) \cup \ldots \cup Var(t_n)$.
- If A and B are formulas, and

  $$BVAR(A) \cap BVAR(B) = FVAR(A) \cap BVAR(B) = BVAR(A) \cap FVAR(B) = \emptyset,$$

  then $A \rightarrow B$ is a formula, and we let $BVAR(A \rightarrow B) = BVAR(A) \cup BVAR(B)$ and $FVAR(A \rightarrow B) = FVAR(A) \cup FVAR(B)$.
- if A is a formula and $x \in FVAR(A)$ then $\forall x A$ is a formula; we let $BVAR(\forall x A) = BVAR(A) \cup \{x\}$ and $FVAR(\forall x A) = FVAR(A) - \{x\}$.

■

[5]By analogy with tableaux (see the previous footnote), one may study more refined skolemization techniques which minimize the free variables on which the skolem function depends [] WE MUST FIND THE REFERENCES.

It is easy to see that for all formulas A, $FVAR(A) \cap BVAR(A) = \emptyset$, and each universal quantifier acts on a different variable, that is we do not allow formulas such as $\forall x(p(x) \rightarrow \forall x p(x))$.

Every formula of the language can be displayed as

$$\forall \bar{x_1}(A_1 \rightarrow \forall \bar{x_2}(A_2 \rightarrow \ldots \forall \bar{x_k}(A_k \rightarrow \forall \bar{y} q) \ldots),$$

where A_i are arbitrary formulas, $\bar{x_i}$ and $\bar{y}$ are (possibly empty) sequences of variables, and q is an atomic formula. According to the definition, variables $\bar{y}$ cannot occur in A_i and variables $\bar{x_i}$ cannot occur in A_j for $j < i$. The restrictions involved in our definition of formulas do not cause any loss of generality, as we can always rename bound variables. A formula A' which is obtained from a formula A by renaming some or all bound variables of A is called a *variant* of A.

Given a formula A we extend the definition of $Body(A)$ and $Head(A)$ formulas of the form $\forall x A$:

$Body(\forall x A) = Body(A)$ and $Head(\forall x A) = Head(A)$.

Thus, in a formula $A = \forall \bar{x_1}(A_1 \rightarrow \forall \bar{x_2}(A_2 \rightarrow \ldots \forall \bar{x_k}(A_k \rightarrow \forall \bar{y}\ q) \ldots)$, he have that

$Body(A) = \{A_1, \ldots, A_k\}$ and $Head(A) = q$.

We assume that the usual notions regarding substitutions (composition, empty substitution, mgu ecc...) are known (the reader is referred to [Lloyd 84]). A substitution may act only on the free variables of a formula, hence if $\theta = \{x/a, y/b\}$, then

$(\forall y p(x, y))\theta = \forall y p(a, y)$.

Given two substitutions σ, γ, we define

$\sigma \leq \gamma$ (σ is an instance of γ) iff there is a substitution δ, such that $\sigma = \gamma\delta$.

As usual a database is a finite set of formulas.

The proof procedure we present below manipulates *queries* N which are finite sets of *basic queries* of the form $(\Delta \ \vdash^? \ A)$, where Δ is database A is a formula. As in conventional logic languages, the proof procedure described below compute *answer substitutions* in case of success.

Definition 2.6.3 A derivation of a query N is a sequence of queries $N_0, N_1, \ldots, N_k$, together with a sequence of substitutions $\sigma_1, \ldots, \sigma_k$, such that $N_0 = N$, and for $i = 0, \ldots, k$ N_{i+1}, and σ_{i+1} are determined according to one of the following rules:

- (Success) if $N_i = N' \cup \{(\Delta \ \vdash^? \ q)\}$, where q is an atom, and there is a formula $C \in \Delta$ and a variant C' of C such that $BVAR(C') \cap FVAR(\Delta \cup \{q\}) = \emptyset$, $Body(C') = \emptyset$, and there exists $\sigma = mgu(Head(C'), q)$, then we can put

$$N_{i+1} = N'\sigma, \quad \sigma_{i+1} = \sigma.$$

- (Implication) if $N_i = N' \cup \{(\Delta \ \vdash^? \ A \rightarrow B)\}$, then we can put

$$N_{i+1} = N' \cup \{(\Delta \cup \{A\} \ \vdash^? \ B)\}, \quad \sigma_{i+1} = \epsilon.$$

- (For all) if $N_i = N' \cup \{(\Delta \ \vdash^? \ \forall A)\}$, $\bar{u} = \{u_1, \ldots, u_n\} = FVAR(\Delta \cup \{A\})$, and f is a function symbol not occurring in $\Delta \cup \{A\}$, then we can put

$$N_{i+1} = N' \cup \{(\Delta \ \vdash^? \ A[x/f(\bar{u})])\}, \quad \sigma_{i+1} = \epsilon.$$

- (Reduction) if $N_i = N' \cup \{(\Delta \ \vdash^? \ q)\}$, where q is an atom, and there is a formula $C \in \Delta$ and a variant C' of C such that $BVAR(C') \cap FVAR(\Delta \cup q) = \emptyset$, $Body(C') = \{A_1, \ldots, A_n\}$, and there exists $\sigma = mgu(Head(C'), q)$, then we can put

$$N_{i+1} = N'\sigma \cup \{(\Delta\sigma \ \vdash^? \ A_1\sigma), \ldots, (\Delta\sigma \ \vdash^? \ A_n\sigma)\}$$

and

$$\sigma_{i+1} = \sigma.$$

A successful derivation of N is a derivation $\mathcal{D} = N_0, \ldots, N_k$, $\sigma_1, \ldots, \sigma_k$, such that $N_k = \emptyset$. The *answer substitution* θ computed by $\mathcal{D}$ is defined as the composition of σ_i restricted to the free- variables of N, that is

$$\theta = (\sigma_1\sigma_2 \ldots \sigma_k)_{|FV(N)}.$$

We conventionally assume that $dom(\theta) = FV(N)$ [6]. We say that N succeeds with answer θ if there is a successful derivation of N computing answer θ. ■

Example 2.6.4 Let Δ be the following set of formulas:

$\forall x(p(x) \to r(x))$
$\forall y(r(y) \to q(y))$
$\forall v(s(v) \to [\forall z p(z) \to \forall u q(u)] \to a(v))$
$\forall w([s(w) \to a(w)] \to t)$

In Fig.2.6, we show a derivation of $\Delta \ \vdash^? \ t$. To lighten notation, we omit the parentheses around each basic query. ■

The following propositions state some basic properties of the computation.

Proposition 2.6.5 *Let N be a query then we have*

1. *if N has a successful derivation of length h with computed answer σ, then $N\sigma$ has a successful derivation of length $\leq h$ with computed answer ϵ;*

2. *if N has a successful derivation of length h with computed σ and $\theta \leq \sigma$, then $N\theta$ has a successful derivation of length $\leq h$ with computed answer ϵ.*

3. *if $N\sigma$ has a successful derivation of length h with computed answer θ, then there is a substitution γ such that $\sigma\theta \leq \gamma$ and N has a successful derivation of length $\leq h$ with computed answer γ;*

4. *if $N = N' \cup \{(\Delta \ \vdash^? \ \forall y A)\}$, and N has a successful derivation of length h with computed σ, then for any term t, the query*

$$N' \cup \{(\Delta \ \vdash^? \ A[y/t])\}$$

has a successful derivation of length $\leq h$ with computed answer σ.

Proof. All claims can be proved by induction on the length of the derivations. We omit the details. □

Proposition 2.6.6 *(Property of monotonicity) If $(\Gamma \ \vdash^? \ A)$ succeeds with answer θ and height h, then $(\Gamma \cup \Delta \ \vdash^? \ A)$ succeeds with answer θ and height $\leq h$.*

[6]This can always be achieved by extending θ with dummy bindings $\{x/x\}$, for variables $x \in FV(N)$ which have no proper bindings in θ.

Proposition 2.6.7 *Given two queries N_1, N_2, we have:*

(1) if N_1 and N_2 both succeed with answer θ, then the query $N_1 \cup N_2$ succeeds with answer θ;

(2) if $N_1 \cup N_2$ succeeds with θ, then there exists $\theta_1, \theta_2 \geq \theta$, such that N_1 succeeds with θ_1 and N_2 succeeds with θ_2.

Proof.

1. Let $\mathcal{D}$ be a successful derivation of N_1, starting from $N_1 \cup N_2$ we perform the same steps as in the $\mathcal{D}$, this will leave us at the end with $N_2\theta$. By hypothesis and proposition 2.6.5(1), $N_2\theta$ succeeds with answer ϵ, thus from the initial query $N_1 \cup N_2$, we get the answer $\theta\epsilon = \theta$.

2. Let $N_1 \cup N_2$ succeeds with θ, then $(N_1 \cup N_2)\theta$ succeeds with ϵ, (proposition 2.6.5(1)); this implies that both $N_1\theta$ and $N_2\theta$ succeed with ϵ; now we can conclude by proposition 2.6.5(3).

□

Proposition 2.6.8 *(Identity) For all Δ, A,*

$$\Delta \cup \{A\} \vdash^{?} A \textit{ succeeds with } \epsilon.$$

In order to prove the completeness of the procedure (proved at the end of the section), we need to show that computation are closed with resoect to the cut-rule, i.e. the cut-rule is adimissible. This property might be of interest by itself. First, we must define what we intend by cut in this context. In order to formulate the cut property, we must remember that the proof-procedure checks the success and simultaneous answers for conjunctions or sets of basic queries. The computed answer must be taken into account. For instance

(1) $p(x) \vdash^{?} p(a)$ succeeds with answer x/a, and
(2) $q(b), \forall z(q(z) \rightarrow p(z)) \vdash^{?} p(x)$ succeeds with x/b,

by cutting (1) and (2) on $p(x)$ we otain

$$q(b), \forall z(q(z) \rightarrow p(z)) \vdash^{?} p(a),$$

which obviously fails. In order to perform a cut the computed answers of the two premises must be *compatible.* We say that two substitution σ and θ are compatible if they have a common instance δ. If

$\Gamma \vdash^{?} A$ succeeds with answer σ and $\Delta, A \vdash^{?} B$ succeeds with answer θ,

and σ and θ are compatible, that is there is a common instance δ σ and θ, we are able to cut on A. We expect that the resulting query $\Gamma, \Delta \vdash^{?} B$ succeeds with a substitution γ which is at least as general as the common instance δ. We need to define cut on queries, which are sets of basic queries. Given a query $N = (\Delta_1 \vdash^{?} B_1), \ldots, (\Delta_n \vdash^{?} B_n)$, a formula A, and a database Γ, we denote by

$$N[A/\Gamma] = (\Delta_1' \vdash^{?} B_1), \ldots, (\Delta_n' \vdash^{?} B_n)$$

the query obtained from N by replacing A in Δ_i by Γ, that is

$$\Delta_i' = \begin{cases} (\Delta_i - \{A\}) \cup \Gamma \text{ if } A \in \Delta_i \\ \Delta_i \text{ otherwise} \end{cases}$$

Theorem 2.6.9 *If the following conditions hold:*

(1) $\Gamma \vdash^? A$ *succeeds with answer* σ,
(2) N *succeeds with answer* θ,
(3) there exist some substitutions ϕ_1, ϕ_2, *such that* $\sigma\phi_1 = \theta\phi_2$,

then there exists a substitution γ *such that* $\theta\phi_2 \leq \gamma$ *and*

$N[A/\Gamma]$ *succeeds with answer* γ.

Proof. By induction on pairs (c, h) where c is the complexity of A, and h is the length of a successful derivation of (2). The complexity $cp(A)$ of a formula A is defined as in chapter 1, with the additional stipulation that $cp(\forall x A) = cp(A) + 1$. Let $c = 0$, we consider the case when $h \geq 0$, and $N \neq \emptyset$. Suppose first $N = N' \cup \{(\Delta \cup \{A\} \vdash^? q)\}$ and success rule is applied to N on $(\Delta \cup \{A\} \vdash^? q)$. This means that from N we step to $N'\pi$, which succeeds with height $< h$, where $\pi = mgu(Head(C'), q)$, for some variant C' of a formula $C \in \Delta$, whose body is empty, and $\theta = \pi\theta'$. If $h = 0$ the proof below simplifies, since $N' = \emptyset$, and we do not need to apply the inductive hypothesis. Let $A = q_1$.

Suppose $C \neq A$ then, since $\sigma\phi_1 = \pi\theta'\phi_2 \leq \sigma$, we have that

$\Gamma\pi\theta'\phi_2 \vdash^? A\pi\theta'\phi_2$ succeeds with ϵ.

By proposition 2.6.5, $\Gamma\pi \vdash^? q_1\pi$ succeeds with some η such that $\theta'\phi_2 \leq \eta$ and height $< h$. Since for some δ, $\theta'\phi_2 = \eta\delta$, by induction hypothesis, we get

$N'\pi[A\pi/\Gamma\pi] = N'[A/\Gamma]\pi$ succeeds with some $\gamma \geq \theta'\phi_2$.

Since $C \neq A$, we obtain, by monotonicity, that

$N'[A/\Gamma] \cup \{(\Delta \cup \Gamma \vdash^? q)\}$ succeeds with $\pi\gamma$, and $\theta\phi_2 = (\pi\theta')\phi_2 \leq \pi\gamma$.

If $C = A = q_1$, let $\pi = mgu(q_1, q)$, then $N'\pi$ succeeds with height $< h$ and $\theta = \pi\theta'$. Let $\alpha = \pi\theta'\phi_2$. Using proposition 2.6.5 we have that

$(*)$ $\Gamma\alpha \vdash^? q_1\alpha$ and $N'\alpha$ both succeeds with ϵ.

Moreover, by the proposition 2.6.5 $N'\alpha$ succeeds with height $< h$. Then we can apply the inductive hypothesis and conclude that

$N'\alpha[q_1/\Gamma\alpha] = N'[q_1/\Gamma]\alpha$ succeeds with ϵ.

Since $q_1\alpha = q\alpha$, we can combine the above conclusion with (*) (proposition 2.6.7), and by monotonicity we have that

$(N'[q_1/\Gamma]\alpha \cup \{(\Delta \cup \Gamma \vdash^? q)\})\alpha$ succeeds with ϵ.

By the proposition we have that for some $\delta \geq \alpha$,

$N'[q_1/\Gamma]\alpha \cup \{(\Delta \cup \Gamma \vdash^? q)\}$ succeeds with answer δ.

But since $\theta\phi_2 = \pi\theta'\phi_2 = \alpha \leq \delta$, we have obtained the desired conclusion.

Next we consider the cases when $c = 0$, $h > 0$, N has the form

$N = N' \cup \{(\Delta \cup \{A\} \ \vdash^? \ B)\}$

and the next query in the derivation is obtained by applying either implication rule or the rule for universal quantification to $(\Delta \cup \{A\} \ \vdash^? \ B)$. Since these two rules do not modify bindings, we can just apply the inductive hypothesis to the next queries and conclude. We omit the details.

We now consider the case when $c = 0$, $h > 0$, N has the form

$N = N' \cup \{(\Delta \cup \{A\} \ \vdash^? \ q)\}$

and the next query in the derivation is obtained by applying reduction to the basic query $(\Delta \cup \{A\} \ \vdash^? \ q)$. Since $cp(A) = c = 0$, q is reduced with respect to $C \in \Delta$ different from A. Then, for some variant C' of C such that $BVAR(C') \cap FVAR(\Delta \cup \{A, q\}) = \emptyset$, $Body(C') = \{D_1, \ldots, D_n\}$, there exists $\pi = mgu(Head(C'), q)$, and we step to

(i) $N'\pi \cup \{(\Delta\pi \cup \{A\pi\} \ \vdash^? \ D_1\pi), \ldots, (\Delta\pi \cup \{A\pi\} \ \vdash^? \ D_n\pi)\}$

which suceeds with θ' such that $\theta = \pi\theta'$, and with height $< h$. Since $\sigma\phi_1 = \pi\theta'\phi_2 \leq \sigma$, we have that

$\Gamma\pi\theta'\phi_2 \ \vdash^? \ A\pi\theta'\phi_2$ succeeds with ϵ.

By proposition 2.6.5,

(ii) $\Gamma\pi \ \vdash^? \ A\pi$ succeeds with some η such that $\theta'\phi_2 \leq \eta$ and height $< h$.

We have that, for some δ, $\theta'\phi_2 = \eta\delta$; we can apply the induction hypothesis to (i) and (ii), and obtain that $N'\pi[A\pi/\Gamma\pi] \cup \{(\Delta\pi \cup \Gamma\pi \ \vdash^? \ D_1\pi), \ldots, (\Delta\pi \cup \Gamma\pi \ \vdash^? \ D_n\pi)\}$ that is

(iii) $N'[A/\Gamma]\pi \cup \{((\Delta \cup \Gamma)\pi \ \vdash^? \ D_1\pi), \ldots, ((\Delta \cup \Gamma)\pi \ \vdash^? \ D_n\pi)\}$ succeeds with some $\gamma \geq \theta'\phi 2$.

Thus, we can reduce the query $N'[A/\Gamma] \cup \{(\Delta \cup \Gamma \ \vdash^? \ q)\}$ to (iii) and succeed with $\pi\gamma$. Since $\theta\phi_2 = (\pi\theta')\phi_2 \leq \pi\gamma$, we have obtained the desired result.

Suppose now that $cp(A) = c > 0$. It is easily seen that there is only one additional case: that one in which $N = N' \cup \{(\Delta \cup \{A\} \ \vdash^? \ q)\}$ and the next query in the derivation is obtained by reduction of q with respect to A. Let

$A = \ \forall \bar{x_1}(D_1 \to \forall \bar{x_2}(D_2 \to \ldots \forall \bar{x_k}(D_k \to \forall \bar{y} q_1) \ldots)$

and let A' be a variant of A such that $BVAR(A') \cap FVAR(\Delta, q) = \emptyset$. Then we have that $Body(A') = \{D'_1, \ldots, D'_k\}$ and $Head(A') = q'_1$. Then, there exists $\pi = mgu(q'_1, q)$ and we step to

(i) $N'\pi \cup \{((\Delta\pi \cup \{A\pi\} \ \vdash^? \ D'_1\pi), \ldots, (\Delta\pi \cup \{A\pi\} \ \vdash^? \ D'_k\pi)\}$

which succeeds with θ' such that $\theta = \pi\theta'$, and with height $< h$. We can proceed as before: since $\sigma\phi_1 = \pi\theta'\phi_2 \leq \sigma$, we have that $\Gamma\pi\theta'\phi_2 \ \vdash^? \ A\pi\theta'\phi_2$ succeeds with ϵ. By proposition 2.6.5,

(ii) $\Gamma\pi \ \vdash^? \ A\pi$ succeeds with some η_1 such that $\theta'\phi_2 \leq \eta_1$ and height $< h$.

We have that, for some δ, $\theta'\phi_2 = \eta_1\delta$; we can apply the induction hypothesis to (i) and (ii), and obtain that

(iii) $N'[A/\Gamma]\pi \cup \{((\Delta \cup \Gamma)\pi \ \vdash^? \ D'_1\pi), \ldots, ((\Delta \cup \Gamma)\pi \ \vdash^? \ D'_k\pi)\}$ succeeds with some $\gamma \geq \theta'\phi 2$.

From (ii), by proposition 2.6.8, we get

$\Gamma\pi \ \vdash^? \ D'_1\pi \rightarrow (\forall \bar{x_2}(D_2 \rightarrow \ldots \forall \bar{x_k}(D_k \rightarrow \forall \bar{y} q_1) \ldots)\pi$ succeeds with η_1, and hence also
$\Gamma\pi \cup \{D'_1\pi\} \ \vdash^? \ (\forall \bar{x_2}(D_2 \rightarrow \ldots \forall \bar{x_k}(D_k \rightarrow \forall \bar{y} q_1) \ldots)\pi$.

We can proceed by proposition 2.6.5 and implication rule and conclude that:

$\Gamma\pi \cup \{D'_1\pi, D'_2\pi \ldots D_k\pi\} \ \vdash^? \ q'_1\pi$ succeeds with η_1.

But since $q'_1\pi = q\pi$, we also get

(iv) $\Gamma\pi \cup \{D'_1\pi, D'_2\pi \ldots D_k\pi\} \ \vdash^? \ q\pi$ succeeds with η_1.

On the other hand from (iii) by propositions 2.6.5 and 2.6.7 we get that

(a) $N'[A/\Gamma]\pi\theta'\phi_2$ succeeds with ϵ,

and for $i = 1, \ldots, k$

(b_i) $(\Delta\pi \cup \Gamma\pi \ \vdash^? \ D'_i\pi)$ succeeds with answer γ_i such that $\theta'\phi_2 \leq \gamma_i$.

We have that for some χ_1 and ψ_1 $\eta_1\chi_1 = \theta'\phi_2 = \gamma_1\psi_1$, Furthermore, it holds that $cp(D'_1) < cp(A)$, we can hence apply the induction hypothesis to (iv) and (b_1) and obtain that:

(v) $\Delta\pi \cup \Gamma\pi \cup \{D'_2\pi \ldots D_k\pi\} \ \vdash^? \ q\pi$ succeeds with some η_2, such that $\theta'\phi_2 \leq \eta_2$.

We can repeat the same argument, now using (v) and (b_2). At the end we obtain that $\Delta\pi \cup \Gamma\pi \ \vdash^? \ q\pi$ succeeds with some η_{k+1} such that $\theta'\phi_2 \leq \eta_{k+1}$. By proposition 2.6.5, we get that

(c) $\Delta\pi\theta'\phi_2 \cup \Gamma\pi\theta'\phi_2 \ \vdash^? \ q\pi\theta'\phi_2$ succeeds with ϵ.

By proposition 2.6.7, we can combine (a) and (c) and conclude that $(N'[A/\Gamma] \cup \{(\Delta \cup \Gamma \ \vdash^? \ q))\}\pi\theta'\phi_2$ succeeds with ϵ. Finally, by proposition 2.6.5, we obtain that for some β such that $\pi\theta'\phi_2 \leq \beta$,

$N'[A/\Gamma] \cup \{(\Delta \cup \Gamma \ \vdash^? \ q))\}$ succeeds with answer β.

Since it holds that $\theta\phi_2 = (\pi\theta')\phi_2 \leq \beta$, we have obtained the desired conclusion. □

Corollary 2.6.10 *(a) If* $\Gamma \ \vdash^? \ A$ *succeeds with* σ *and* $\Delta \cup \{A\} \ \vdash^? \ B$ *succeeds with* θ *and for some substitutions* $\phi_1, \phi 2$ $\sigma\phi_1 = \theta\phi_2$*, then there is a substitution* $\gamma \geq \theta\phi_2$ *such that*

$\Delta \cup \Gamma \ \vdash^? \ B$ *succeeds with* γ*.*

(b) In particular, if $\Gamma \ \vdash^? \ A$ *and* $\Delta \cup \{A\} \ \vdash^? \ B$ *both succeed with* ϵ *then* $\Delta, \Gamma \ \vdash^? \ B$ *succeeds with* ϵ*.*

We now prove the soundness and completeness of the the proof procedure. To this aim we introduce the Kripke semantics for the first-order fragment $\mathcal{L}(\forall, \rightarrow, \wedge)$ of intuitionistic logic, which is sufficient to interpret our queries. For the fragment we deal with, it is sufficient to consider *constant domain* Kripke models. We also assume that the denotations of terms are rigid, i.e. do not depend on worlds.

Definition 2.6.11 A structure M for a language $\mathcal{L}(\forall, \rightarrow, \wedge)$ is a quadruple $M = (W, D, I \leq)$ where D and W are non empty sets, $\leq$ is a reflexive-transitive relation, and I, (called the interpretation function), maps

- every n-ary function symbol f on a n-ary function $I(f) : D^n \rightarrow D$,
- every n-ary predicate symbol p on a function $I(p) : W \rightarrow 2^{D_n}$, that is $I(p)(w)$ for $w \in W$ is an n-ary relation on W,
- every variable x on an element of D.

I is assumed to be increasing on the interpretation of predicates:

$$w \leq w' \;\Rightarrow\; I(w)(p) \subseteq I(w')(p).$$

The interpretation of terms is defined as in first-order classical structures. To define truth in a structure, we assume that the language is expanded with names (constants) for all elements of D; we will not distinguish between an element of D and its name. We define $M, w \models A$, which is read as "A is true in M at world w", as follows:

$M, w \models p(\bar{t})$ iff $\bar{t} \in I(p)(w)$;
$M, w \models A \wedge B$ iff $M, w \models A$ and $M, w \models B$;
$M, w \models A \rightarrow B$ iff for all w' such that $w \leq w'$ $M, w' \models A$ implies$M, w' \models B$;
$M, w \models \forall x A$ iff $\forall d \in D$ $M, w \models A[x/d]$[7]

Validity in a structure M (i.e. $M \models A$) is defined as truth in every world of M, and *logical validity* (i.e. $\models_{\mathbf{I}} A$) is defined as validity in any structure. ■

HERE I HAVE DEFINED MODELS WITHOUT A LEAST WORLD, DO WE CHANGE THE DEFINITION? IS TRUTH IN THE LEAST WORLD EQUIVALENT TO TRUTH IN EVERY WORLD???

The next theorem states the soundness of the procedure with respect to intuitionistic logic.

Theorem 2.6.12 (Soundness) *Let $N = (\Delta_1 \;\vdash^?\; A_1), \ldots, (\Delta_n \;\vdash^?\; A_n)$ be a query. If N succeeds with answer θ, then $\forall\big[\bigwedge_i^n (\bigwedge \Delta_i \rightarrow A_i)\theta\big]$ is valid in intuitionistic logic.*

Proof. By induction on the height of a successful derivation of N. We omit the details. □

We prove the completeness of the procedure by a canonical model construction which makes an essential use of the cut-admissibility theorem 2.6.9.

Canonical Model construction
We consider a language $\mathcal{L}$ which contains infinitely many function symbols of each-arity, and we define a structure $M = (W, D, \subseteq, I)$, where W is the set of databases (finite sets of formulas) on $\mathcal{L}$, D is the set of terms on $\mathcal{L}$, and I is the identity on terms, and for every $\Delta \in W$, and predicate p

$$\bar{t} \in I(p)(\Delta) \;\Leftrightarrow\; \Delta \;\vdash^?\; p(\bar{t}) \text{ succeeds with } \epsilon.$$

By monotonicity of deduction it is easily seen that I is increasing on the interpretation of predicates.

Proposition 2.6.13 *For any Δ, A we have:*

$$M, \Delta \models A \;\Leftrightarrow\; \Delta \;\vdash^?\; A \textit{ succeeds with } \epsilon.$$

[7]Notice that, in the case of $\forall$ we do not need to consider truth in upper worlds; this because we are dealing with constant domain Kripke models.

Proof. We prove the two directions simultaneously by induction on the complexity of A. If A is an atomic formula, the claim holds by definition of I.

Let $A = B \rightarrow C$.

($\Rightarrow$) Suppose that $M, \Delta \models B \rightarrow C$. By identity we have that $\Delta \cup \{B\} \ \vdash^? \ B$ succeeds with ϵ. Let $\Gamma = \Delta \cup \{B\}$, by induction hypothesis we have $M, \Gamma \models B$; since $\Delta \subseteq \Gamma$, by hypothesis we can conclude that $M, \Gamma \models C$, so that by induction hypothesis $\Delta \cup \{B\} \ \vdash^? \ C$ succeeds with ϵ; this implies that $\Delta \ \vdash^? \ B \rightarrow C$ succeeds with ϵ.

($\Leftarrow$) Suppose that $\Delta \ \vdash^? \ B \rightarrow C$ succeeds with ϵ. By implication rule, it must be that

(i) $\Delta \cup \{B\} \ \vdash^? \ C$ succeeds with ϵ.

Now let Γ be a database such that $\Delta \subseteq \Gamma$ and $M, \Gamma \models B$. By induction hypothesis we get that

(ii) $\Gamma \ \vdash^? \ B$ succeeds with ϵ.

From (i) and (ii) by cut (corollary 2.6.10), we obtain $\Gamma \cup \Delta \ \vdash^? \ C$ succeeds with ϵ, that is the same as $\Gamma \ \vdash^? \ C$ succeeds with ϵ, since $\Delta \subseteq \Gamma$. By induction hypothesis we finally get $M, \Gamma \models C$.

Let $A = \forall x B$.

($\Rightarrow$) Suppose that $M, \Delta \models \forall x B$, then, for any term t we have that $M, \Delta \models B[x/t]$. Let $\{u_1, \dots u_k\} = \bar{u} = FV(\Delta \cup \{B\})$ and let f be a k-ary function symbol not occurring in $\Delta \cup \{B\}$, then we have in particular that $M, \Delta \models B[x/f(\bar{u})]$, and by induction hypothesis we get that $\Delta \ \vdash^? \ B[x/f(\bar{u})]$ succeeds with ϵ. By the for-all rule this implies that also $\Delta \ \vdash^? \ \forall x B$ succeeds with ϵ.

($\Leftarrow$) Suppose that $\Delta \ \vdash^? \ \forall B$ succeeds with ϵ; by proposition 2.6.5, we have that for every term t, $\Delta \ \vdash^? \ B[x/t]$ succeeds with ϵ. By induction hypothesis we get that, for all t $M, \Delta \models B[x/t]$ and hence $M, \Delta \models \forall x B$. □

From theorem 2.6.13 we easily obtain the completeness of the proof procedure. As in conventional Horn logic programming, we cannot expect that for every θ, if $\models_{\mathbf{I}} (\Delta \rightarrow A)\theta$, then $\Delta \ \vdash^? \ A$ succeeds with θ, but only that there is some substitution γ at least as general as θ such that $\Delta \ \vdash^? \ A$ succeeds with γ. For instance, for any θ, $(p(x) \rightarrow p(x))\theta$ is valid, but $\emptyset \ \vdash^? \ p(x) \rightarrow p(x)$ succeeds with ϵ only.

Theorem 2.6.14 (Completeness) *Let $N = (\Delta_1 \ \vdash^? \ A_1), \dots, (\Delta_n \ \vdash^? \ A_n)$ be a query. If $\forall \big[\bigwedge_i^n (\bigwedge \Delta_i \rightarrow A_i)\theta \big]$ is valid in intuitionistic logic, then $N\theta$ succeeds with ϵ and N succeeds with γ such that $\theta \leq \gamma$.*

Proof. Suppose that $\forall \big[\bigwedge_i^n (\bigwedge \Delta_i \rightarrow A_i)\theta \big]$ is valid then it is valid in the canonical model M defined above. Thus, in each world Γ of M, we have $M, \Gamma \models \bigwedge_i^n (\bigwedge \Delta_i \rightarrow A_i)\theta$. Since for every i, $\Delta_i\theta \in W$, so that, in particular, $M, \Delta_i\theta \models \bigwedge \Delta_i\theta \rightarrow A_i\theta$. Since $M, \Delta_i\theta \models \bigwedge \Delta_i\theta$, we get that $M, \Delta_i\theta \models A_i\theta$. By the previous proposition, we obtain that

$\Delta_i\theta \ \vdash^? \ A_i\theta$ succeeds with ϵ, for $i = 1, \dots, n$.

This implies that $N\theta$ succeeds with ϵ and N succeeds with γ such that $\theta \leq \gamma$ by proposition 2.6.5. □

$$\begin{tikzcd}[column sep=small]
 & \vdash^{?}\ x : a \to t \vee s \arrow[d, dash] & & \\
 & x \leq y, y : a \vdash^{?}\ y : t \vee s \arrow[d, dash] & & \\
 & \vdash^{?}\ y : t,\ (y : s) \arrow[d, dash] & & \\
 & \vdash^{?}\ y : b \to p \arrow[d, dash] & & \\
 & y \leq z, z : b \vdash^{?}\ z : p \arrow[d, dash] & & \\
 & (*)\ \ \vdash^{?}\ z : c \arrow[dl, dash] \arrow[dr, dash] & & \\
(i)\ \ \vdash^{?}\ y : a & & y : (b \to c) \vee (d \to e) \vdash^{?}\ z : c \arrow[dl, dash] \arrow[dr, dash] & \\
 & y : b \to c \vdash^{?}\ z : c \arrow[d, dash] & & (iii)\ y : d \to e \vdash^{?}\ z : c \arrow[d, dash] \\
 & (ii)\ y : b \to c \vdash^{?}\ z : b & & \vdash^{?}\ y : s \text{ (restart)} \arrow[d, dash] \\
 & & & \vdash^{?}\ y : q \to e \arrow[d, dash] \\
 & & & y \leq u, u : q \vdash^{?}\ u : e \arrow[d, dash] \\
 & & & \vdash^{?}\ u : d \arrow[d, dash] \\
 & & & \vdash^{?}\ u : q
\end{tikzcd}$$

Figure 2.3:

$$\begin{tikzcd}
 & \Delta \vdash x : (b \to a) \vee c, \emptyset \arrow[d, dash] & \\
 & \Delta \vdash x : b \to a, \{(x,c)\} \arrow[d, dash] & \\
 & x \leq y, \Delta, y : b \vdash y : a, \{(x,c)\} \arrow[dl, dash] \arrow[dr, dash] & \\
x \leq y, \Delta, y : b, x : a \vdash y : a, \{(x,c)\} & & x \leq y, \Delta, y : b, x : b \to c \vdash y : a, \{(x,c)\} \arrow[d, dash] \\
 & & x \leq y, \Delta, y : b, x : b \to c \vdash x : c, \{(x,c),(y,a)\} \text{ restart} \arrow[d, dash] \\
 & & (*) x \leq y, \Delta, \mathbf{y} : b, x : b \to c \vdash \mathbf{x} : b, \{(x,c),(y,a)\}
\end{tikzcd}$$

Figure 2.4:

$$\begin{tikzcd}
\emptyset \ \vdash^{?} \ \forall x((p(x) \to \forall y p(y)) \to q) \to q \arrow[d, dash] \\
\forall x((p(x) \to \forall y p(y)) \to q) \ \vdash^{?} \ q \arrow[d, dash] \\
\forall x((p(x) \to \forall y p(y)) \to q) \ \vdash^{?} \ p(x) \to \forall y p(y) \arrow[d, dash] \\
\forall x((p(x) \to \forall y p(y)) \to q), p(x) \ \vdash^{?} \ \forall y p(y) \arrow[d, dash] \\
\forall x((p(x) \to \forall y p(y)) \to q), p(x) \ \vdash^{?} \ p(c)
\end{tikzcd}$$

Figure 2.5:

$$\begin{array}{c}
\Delta \ \vdash^{?} \ t \\
| \\
\Delta \ \vdash^{?} \ s(w') \to a(w') \\
| \\
\Delta, s(w') \ \vdash^{?} \ a(w') \\
| \\
\Delta s(w') \ \vdash^{?} \ s(w'), ((\Delta, s(w) \ \vdash^{?} \ \forall p(z) \to \forall u q(u)) \\
| \\
\Delta, s(w'), \forall z p(z) \ \vdash^{?} \ \forall u q(u) \\
| \\
\Delta, s(w'), \forall z p(z) \ \vdash^{?} \ q(f(w')) \\
| \\
\Delta, s(w'), \forall z p(z) \ \vdash^{?} \ r(f(w')) \\
| \\
\Delta, s(w'), \forall z p(z) \ \vdash^{?} \ p(f(w')) \\
| \\
\text{success, as } mgu(p(z), p(f(w'))) = \{z/f(w')\}
\end{array}$$

Figure 2.6:

Chapter 3

Intermediate logics

We have seen in the previous chapter that intuitionistic logic is complete with respect to the class of finite Kripke models. Given a Kripke model $M = (S, \leq, a, V)$, we can consider the structure $(S, \leq, a)$, which is a finite tree, forgetting about the evaluation function V for atoms. Changing the terminology a bit, we will speak about models based on a given finite tree $(S, \leq, a)$, since varying V we will have several models based on it. The completeness result can be restated as follows: intuitionistic logic is complete with respect to the class of finite trees, that is to say, with respect to Kripke models based on the class of finite trees. This change of terminology matters as we can consider subclasses of finite trees and ask what axioms we can add to obtain a characterization of valid formulas in these subclasses. Let us consider two such subclasses.

(1) For any k, the class of finite trees of height $\leq k$; a finite tree $T = (S, \leq, a)$ is in this class if there are not $k+1$ different elements $a, t_1, \ldots, t_k$, such that

$a \leq t_1 \leq \ldots \leq t_k$ holds.

This means that all chains are of length $\leq k$. Valid formulas in these subclasses are axiomatized by the axioms BH_k of the next section.

(2) For any k, the class of finite trees of width $\leq k$; a finite tree $T = (S, \leq, a)$ is in this class if there are not $k+1$ different elements which are pairwise incomparable. Valid formulas in these subclasses are axiomatized by taking the axiom schema

$(A_1 \rightarrow A_2) \vee (A_2 \rightarrow A_3) \vee \ldots \vee (A_k \rightarrow A_1)$.

for finite width trees of width $\leq k$. Notice that for $k = 2$, we have the axiom

$(A_1 \rightarrow A_2) \vee (A_2 \rightarrow A_1)$

which gives the well known logic **LC** introduced independently by Dummett [Dummett 59] and Gödel REFERENCE??? that we have already mentioned in the previous chapter. This axiom corresponds to the property of linearity: the models are finite ordered sequence of worlds.

In this chapter, we begin the study of these classes of intermediate logics, and we give a goal-directed formulation for the logics of bounded height Kripke models and for Dummett logic **LC** .

3.1 Intermediate logics of bounded height Kripke models

We treat here the sequence of intermediate logics between intuitionistic logic and classical logic of bounded height Kripke models. To formalize logics them, we consider the following axioms

$BH_1 = ((A_1 \to A_0) \to A_1) \to A_1$
$\vdots$
$BH_n = ((A_n \to A_{n-1}) \to A_n) \to A_n.$

where A_{n-1} is an instance of BH_{n-1}. Axiom schema BH_1 is nothing else than Peirce's axiom, thus adding it to intuitionistic logic we get classical logic. The axiom BH_{n+1} is strictly weaker than BH_n. Let $\mathbf{BH}_n$ denote the logic obtained by adding to intuitionistic logic $\mathbf{I}$ the axiom schema BH_n. Then we have the following facts.

Proposition 3.1.1 (REFERENCE???) *(a)* $\mathbf{BH_n} \subset \mathbf{BH_{n+1}}$*;*
(b) $\mathbf{I} = \bigcap_{n=1}^{\omega} \mathbf{BH_n}$*;*
(c) A is a theorem of $\mathbf{BH}_n$ iff A is valid in all models of height $\leq n$.

The proof systems for $\mathbf{BH}_n$ are a simple extension of the labelled procedure we have seen in chapter 2 for intuitionistic logic with disjunction. For brevity, we give it here for the pure implicational fragment, but it is possible to extend it to the full propositional language exactly as shown in definition 2.5.4. We recall the rules of the basic implicational system.

Definition 3.1.2 A query has the form $\Gamma \vdash^?_n x : A, H$, where Γ is a labelled set of formulas and constraints of the form $x \leq y$ where x, y are labels. H, called the history, is a set of pairs of the form

$$\{(x_1, q_1), \ldots, (x_n, q_n)\}$$

where x_i are labels and q_i are atoms. The rules for proving constraints are as follows:

- (≤ 1) $\Gamma \vdash x \leq x$;
- (≤ 2) $\Gamma \vdash x \leq y$, if $x \leq y \in \Gamma$;
- (≤ 3) $\Gamma \vdash x \leq y$, if for some z, $\Gamma \vdash x \leq z$ and $\Gamma \vdash z \leq y$.

For each $n \geq 1$, we define the proof system $\vdash^?_n$ as follows. The logical rules are the following

- (success) $\Gamma \vdash^?_n x : q, H$ immediately succeeds if $y : q \in \Gamma$ and $\Gamma \vdash y \leq x$;
- (implication) from $\Gamma \vdash^?_n x : A \to B, H$ we step to

  $\Gamma, y : A, x \leq y \vdash^?_n y : B, H,$

  where $y \notin Lab(\Gamma) \cup Lab(H) \cup \{x\}$.
- (reduction) from $\Gamma \vdash^?_n x : q, H$ if there is a formula $y : C \in \Gamma$, with $C :\ A_1 \to \ldots \to A_k \to q$ such that $\Gamma \vdash y \leq x$ we step to

  $\Gamma \vdash^?_n x : A_i, H \cup \{(x, q)\}$, for $i = 1, \ldots, k$.

- (n-shifting restart) from $\Gamma \vdash_n x : r, H$ if there are $(x_1, q_1), \ldots, (x_k, q_n) \in H$, such that

  $$\Gamma \vdash x_1 \leq x_2 \ldots \Gamma \vdash x_n \leq x$$

  we step to

  $$\begin{array}{l} \Gamma \vdash x_2 : q_1, H \cup \{(x, r)\}, \\ \vdots \\ \Gamma \vdash x_n : q_{n-1}, H \cup \{(x, r)\}, \\ \Gamma \vdash x : q_n, H \cup \{(x, r)\}. \end{array}$$

■

We have called the last rule "shifting" restart as each goal q_i is re-asked from a point x_{i+1} which is, so to say, above the point x_i from which it was asked the first time (the condition is $x_i \leq x_{i+1}$); the goal q_i is hence shifted from x_i to x_{i+1}. Notice that in case of $n = 1$, the $1 - shifting$-restart rule just prescribes to "shift" a previous goal to the current point, that is it prescribes to reask any previous atomic goal from the current database. The $1 - shifting$-restart is therefore restart for classical logic. If we want a proof system for the whole propositional language we must add the rules of section 2.5, in this extension the histories will contain arbitrary formulas rather than atoms.

Example 3.1.3 In Fig. 3.1, we show a derivation of the following atomic instance of BH_2

$$((a_2 \to ((a_1 \to a_0) \to a_1) \to a_1) \to a_2) \to a_2$$

Database Δ and Δ' are respectively as follows:

$$\begin{array}{l} \Delta = \{x_1 : ((a_2 \to ((a_1 \to a_0) \to a_1) \to a_1) \to a_2\}, \\ \Delta' = \Delta \cup \{x_1 \leq x_2, x_2 \leq x_3, x_3 \leq x_4, x_2 : a_2, x_3 : (a_1 \to a_0) \to a_1, x_4 : a_1\}. \end{array}$$

At step (*) we apply the special 2-shifting-restart rule (i.e. the one for BH$_2$). At this step, the history is $\{(x_1, a_2), (x_3, a_1)\}$ and the current position is x_4, thus, by restart, we "shift" a_2 to x_3 and a_1 to x_4. The queries on the leaves of the tree immediately succeed. ■

We now prove the soundness and completeness of this family of goal-directed procedures with respect to $\mathbf{BH}_n$. We recall the notion of realization we have already introduced in section 2.5.

Definition 3.1.4 Given a Kripke structure $M = (W, w_0, \leq, V)$, a realization of a database Γ in M is a mapping $\rho : \mathcal{A} \to W$, such that

- $x \leq y \in \Gamma$ implies $\rho(x) \leq \rho(y)$;
- $x : A \in \Gamma$ implies $M, \rho(x) \models A$.

Given a query $Q = \Gamma \vdash x : G, H$, we say that Q is valid in $\mathbf{BH}_n$ iff for all M and all realization ρ of Γ in M, we have:

either $M, \rho(x) \models G$, or for some $(y, r) \in H$, $M, \rho(y) \models r$.

■

$$\begin{tikzcd}
 & \vdash_2 \ x_0 : ((a_2 \to ((a_1 \to a_0) \to a_1) \to a_1) \to a_2) \to a_2,\ \emptyset \arrow[d, no head] & \\
 & x_1 : ((a_2 \to ((a_1 \to a_0) \to a_1) \to a_1) \to a_2 \vdash_2 \ x_1 : a_2,\ \emptyset \arrow[d, no head] & \\
 & \Delta \vdash_2 \ x_1 : a_2 \to ((a_1 \to a_0) \to a_1) \to a_1,\ \{(x_1, a_2)\} \arrow[d, no head] & \\
 & \Delta, x_1 \leq x_2, x_2 : a_2 \vdash_2 \ x_2 : ((a_1 \to a_0) \to a_1) \to a_1,\ \{(x_1, a_2)\} \arrow[d, no head] & \\
 & \Delta, x_1 \leq x_2, x_2 \leq x_3, x_2 : a_2, x_3 : (a_1 \to a_0) \to a_1 \vdash_2 \ x_3 : a_1,\ \{(x_1, a_2)\} \arrow[d, no head] & \\
 & \Delta, x_1 \leq x_2, x_2 \leq x_3, x_2 : a_2, x_3 : (a_1 \to a_0) \to a_1 \vdash_2 \ x_3 : a_1 \to a_0,\ \{(x_1, a_2), (x_3, a_1)\} \arrow[d, no head] & \\
 & (*)\Delta, x_1 \leq x_2, x_2 \leq x_3, x_3 \leq x_4, x_2 : a_2, x_3 : (a_1 \to a_0) \to a_1, x_4 : a_1 \vdash_2 \ x_4 : a_0,\ \{(x_1, a_2), (x_3, a_1)\} \arrow[dl, no head] \arrow[dr, no head] & \\
\Delta' \vdash_2 \ x_3 : a_2,\ \{(x_1, a_2), (x_3, a_1)\} & & \Delta' \vdash_2 \ x_4 : a_1,\ \{(x_1, a_2), (x_3, a_1)\}
\end{tikzcd}$$

Figure 3.1:

Theorem 3.1.5 (Soundness) *If $\Delta \vdash^?_n G, H$ succeeds then it is valid in BH_n.*

Proof. We proceed by induction on the derivation of the query in the hypothesis. We only consider the case when the query $\Delta \vdash^?_n x : G, H$ succeeds by n-shifting restart. In this case G is an atom q and there are $(x_1, q_1), \ldots, (x_n, q_n) \in H$, such that

$\Gamma \vdash x_1 \leq x_2 \ldots \Gamma \vdash x_n \leq x,$

and from (*) $\Gamma \vdash^?_n x : r, H$ we step to

$(h_1)\ \Gamma \vdash x_2 : q_1, H \cup \{(x, r)\},$
$\vdots$
$(h_{n-1})\ \Gamma \vdash x_n : q_{n-1}, H \cup \{(x, r)\},$
$(n_n)\ \Gamma \vdash x : q_n, H \cup \{(x, r)\}.$

Since each of the queries above succeeds by a shorter derivation, we can apply the induction hypothesis and assume that $(h_1) - -(h_n)$ are valid. Suppose now that (*) is not valid in $\mathbf{BH_n}$, then for some model $M = (W, w, \leq, V)$ and some realization ρ of Γ in M, we have

(1) $M, \rho(x_{n+1}) \not\models r$, where $x_{n+1} = x$,
(2) for all $(y, q) \in H,\ M, \rho(y) \not\models q$,
(3) $\rho(x_i) \leq \rho(x_{i+1})$ for $i = 1, \ldots, n$.

In particular from (2) we have that

(4) $\ M, \rho(x_i) \not\models q_i$ for every $i = 1, \ldots, n$.

On the other hand by the validity of $(h_1) - -(h_n)$, (1) and (4), we must have that

(5) $M, \rho(x_{i+1}) \models q_i$ for every $i = 1, \ldots, n$.

By (3) we have $\rho(x_1) \leq \rho(x_2) \leq \ldots \leq \rho(x_n) \leq \rho(x_{n+1})$, but by proposition 3.1.1 every chain in M as length at most n, thus it must be

(***) $\rho(x_i) = \rho(x_{i+1})$ for some x_i, with $i = 1, \ldots, n$.

By (4) and (5) we have a contradiction. □

We now turn to completeness, we first notice that for the axiomatization of $\mathbf{BH}_n$, it is sufficient to add to the Hilbert system for intuitionistic logic only the atomic instances of BH_n. To this regard let Π_n be the set of atomic instances of $\mathrm{BH_n}$.

Lemma 3.1.6 (REFERENCES???) *We have*

$\models_{\mathbf{BH_n}} A$ *iff* $\Pi_n \vdash A$ *in intuitionistic logic.*

Proposition 3.1.7 *Let* $\pi_n \in \Pi_n$*, the query*

$\vdash^?_n \ x_0 : \pi_n, \ \emptyset$ *succeeds in the proof system for* $\mathbf{BH}_n$.

Proof. We can assume that $\pi_n = ((q_n \to \pi_{n-1}) \to q_n) \to q_n$, where $q_0, q_1, \ldots, q_n$ are distinct propositional variables, and $\pi_i = ((q_i \to \pi_{i-1}) \to q_i) \to q_i \in \Pi_i$, for $i = 1, \ldots, n$.

We show a derivation of the above query:

$\vdash^?_n \ x_0 : \pi_n, \ \emptyset$
$x_1 : (q_n \to \pi_{n-1}) \to q_n, x_0 \leq x_1 \ \vdash^?_n \ x_1 : q_n, \ \emptyset,$
$x_1 : (q_n \to \pi_{n-1}) \to q_n, x_0 \leq x_1 \ \vdash^?_n \ x_1 : q_n \to \pi_{n-1}, \ \{(x_1 : q_n)\},$
and after two implication steps
$x_1 : (q_n \to \pi_{n-1}) \to q_n, x_2 : q_1, x_3 : (q_{n-1} \to \pi_{n-2}) \to q_{n-1}, x_0 \leq x_1, x_1 \leq x_2, x_2 \leq x_3 \ \vdash^?_n$
$x_3 : q_{n-1}, \ \{(x_1 : q_n)\}$
$x_1 : (q_n \to \pi_{n-1}) \to q_n, x_2 : q_1, x_3 : (q_{n-1} \to \pi_{n-2}) \to q_{n-1}, x_0 \leq x_1, x_1 \leq x_2, x_2 \leq x_3 \ \vdash^?_n$
$x_3 : q_{n-1} \to \pi_{n-2}, \ \{(x_1 : q_n), (x_3 : q_{n-1})\}$

Proceeding this way we arrive to

$\Delta \ \vdash^?_n \ x_{2n} : q_0, H,$

where

$H = \{(x_1 : q_n), (x_3 : q_{n-1}), \ldots, (x_{2n-1}, q_1)\}$ and
$\Delta = \{x_{2(i+1)} : q_{n-i}, x_{2i+1} : (q_{n-i} \to \pi_{n-(i+1)}) \to q_{n-i}, x_i \leq x_{i+1} \mid i = 1, \ldots, n-1\}$

Since $\Delta \vdash x_1 \leq x_3 \ldots \Delta \vdash x_{2n-1} \leq x_{2n}$, we can apply n-shifting-restart and step to

$\Delta \ \vdash^?_n \ x_3 : q_n, H \cup \{(x_{2n}, q_0)\}$
$\Delta \ \vdash^?_n \ x_5 : q_{n-1}, H \cup \{(x_{2n}, q_0)\}$
$\vdots$
$\Delta \ \vdash^?_n \ x_{2n} : q_1, H \cup \{(x_{2n}, q_0)\}$

Each one of the above queries succeeds by the success rule. □

The completeness of the procedure is an easy consequence of the admissibility of cut.

Theorem 3.1.8 (Closure under Cut) *If* $\Gamma[x : D] \vdash^?_n y : D_1, H_1$ *and* $\Delta \vdash^?_n z : D, H_2$ *succeed,then also*

$$\Gamma[x : D/\Delta, z] \vdash^?_n y[x/z] : D_1, H_1[x/z] \cup H_2 \textit{ succeed,}$$

where $\Gamma[x : D/\Delta, z] = (\Gamma - \{x : D\})[x/z] \cup \Delta$.

Proof. As in the previous cases, we proceed by double induction on the complexity of the cut formula and on the length of the derivation. The proof is very similar to the one of theorem 2.5.9 and details are left to the reader. □

Theorem 3.1.9 (Completeness) *Let* A *be valid in* $\mathbf{BH_n}$*, then* $\vdash^?_n x : A, \emptyset$ *for any label* x *succeeds.*

Proof. By proposition 3.1.6 if A is valid in $\mathbf{BH_n}$, then $\Pi'_n \vdash_i A$ holds, where Π'_n is a finite set of atomic instances of Π_n, by the completeness of the proof procedure for intuitionistic logic, we have that $\Pi_n \vdash^? A$ succeeds. The proof system for $\mathbf{BH_n}$ is an extension of the intuitionistic one. It is easy to see that letting $\Pi'_n = \{x_0 : B \mid B \in \Pi_n\}$, we have

(i) $\Pi'_n \vdash^?_n x_0 : A, \emptyset$ succeeds.

But by proposition 3.1.7, we also have that

(ii) $\vdash^?_n x_0 : B, \emptyset$ succeeds for every $B \in \Pi'_n$.

Since the computation are closed under cut (3.1.8), From (i) and (ii) we get that

$\vdash^?_n x_0 : A, \emptyset$ succeeds.

□

3.2 Dummett-Gödel logic LC

In this section we show how we can give a goal-directed procedure for one of the most important intermediate logics, namely Dummett logic **LC** [Dummett 59]. The system **LC** extends intuitionistic logic by the axiom

$$(A \to B) \vee (B \to A).$$

Semantically, this axiom restricts the class of Kripke models of intuitionistic logic, to those which are linearly ordered. This logical system was previously introduced by Gödel [**?**] to show that intuitionistic logic does not admit a characteristic finite matrix.

IS IT RIGHT THIS HISTORICAL REMARK?

Gödel formulated a particularly simple many-valued semantic for **LC** , that we recall below. Formulas are interpreted in the real interval $[0, 1]$; let $v : Var \to [0, 1]$ be an evaluation of the propositional variables, v can be extended to any formula in the language $(\to, \wedge, \neg, \vee)$ according to the following clauses:

$$\begin{aligned}
v(A \to B) &= \begin{cases} 1 \text{ if } v(A) \leq v(B) \\ v(B) \text{ otherwise} \end{cases} \\
v(A \wedge B) &= min(v(A), v(B)) \\
v(A \vee B) &= max(v(A), v(B)) \\
v(\neg A) &= \begin{cases} 1 \text{ if } v(A) = 0 \\ 0 \text{ otherwise} \end{cases}
\end{aligned}$$

We say that a formula A is valid if $v(A) = 1$ under any v on $[0,1]$. Because of its many-valued semantic, logic **LC** is nowadays considered as one of the fundamental *fuzzy logics* (see [Hajek 98]).

An alternative axiomatization of **LC** can be given by adding to the propositional fragment of intuitionistic logic one of the following:

$$(A \to B \vee C) \to (A \to B) \vee (A \to C)$$

$$(A \wedge B \to C) \to (A \to C) \vee (B \to C).$$

The first axiom allows one to carry out a disjunction from the consequent of an implication. It is easily seen that the converse of each axiom above is provable in intuitionistic logic.

The implicational frag ment of **LC** can be axiomatized by adding to the implicational fragment of intuitionistic logic the following axiom:

$$((A \to B) \to C) \to ((B \to A) \to C) \to C.$$

We can formulate a proof procedure for **LC** similar to the one for systems $\mathbf{BH}_n$ of the previous section, which makes use of labelled databases and constraints. To this purpose we replace shifting restart by the following two rules

- (backtracking) from $\Gamma \vdash^? x : q, H$, with q atomic if $(y, r) \in H$ step to

  $\Delta \vdash^? y : r, H \cup \{(x, q)\}$.

- (linear restart) from $\Gamma \vdash^? x : q, H$, with q atomic if $(y, r) \in H$ step to

  $\Gamma, y \leq x \vdash^? x : q, H$ and $\Gamma, x \leq y \vdash^? y : r, H$

The proof system can be extended to the full propositional language as shown in section 2.5, in this case the history of a query will contain arbitrary formulas rather than atoms. The linear restart rule performs a restricted form of case analysis enforced by linearity. We give an example of computation when we have disjunction handled as in section 2.5.

Example 3.2.1 We show (figure 3.2) that $\vdash x : (a \to b) \vee (b \to a), \emptyset$ succeeds. Step (1) is obtained by the rule for disjunction, step (3) by the backtracking rule. We have abbreviated $\{(x : b \to a), (y : b)\}$ by H. Finally steps (5) and (6) are obtained by the linear restart rule, as $(y, b) \in H$.

■

The above procedure can be shown to be sound and complete for **LC** . To this regard we can develop a syntactic proof as we did in the previous section for $\mathbf{BH}_n$ in three steps: (1) for the axiomatization of **LC** , it is enough to add the atomic instances of the axiom $(A \to B) \vee (B \to A)$ to **I** ;

$$\begin{tikzcd}
 & \vdash^? x : (a \to b) \vee (b \to a) \arrow[d, dash] & \\
 & (1)\ \vdash^? x : a \to b, \{(x : b \to a)\} \arrow[d, dash] & \\
 & (2)\ x \leq y, y : a \vdash^? y : b, \{(x : b \to a)\} \arrow[d, dash] & \\
 & (3)\ x \leq y, y : a \vdash^? x : b \to a, \{(x : b \to a), (y : b)\} \arrow[d, dash] & \\
 & (4)\ x \leq y, x \leq z, y : a, z : b \vdash^? z : a, H \arrow[dl, dash] \arrow[dr, dash] & \\
(5)\ x \leq y, x \leq z, \mathbf{y} \leq \mathbf{z}, y : a, z : b \vdash^? z : a, H & & (6)\ x \leq y, x \leq z, \mathbf{z} \leq \mathbf{y}, y : a, z : b \vdash^? y : b, H
\end{tikzcd}$$

Figure 3.2:

(2) every such atomic instance succeeds in the proof system for **LC** , (3) derivations are closed under cut. We leave the details to the reader. We will rather show that we can get rid of labels and obtain a simple procedure for the implicational fragment of **LC** . The unlabelled procedure we define in the next section has a strong relation with the hypersequent calculus for **LC** developed by Avron [Avron 91a], as we will show at the end of the chapter.

I AM NOT SURE THAT THE CLAIM (1) ABOVE IS TRUE, IS IT???

We give an intuitive argument which shows that we can eliminate the labels. To this regard, let us define for any labelled database Γ and label x.

$$\Gamma x = \{y : A \mid y : A \in \Gamma \wedge y \leq x\} \cup \{u \leq v \mid u \leq v \in \Gamma \wedge v \leq x\}$$

Γx represent the set of formulas of Γ which can be used from point x. Moreover, if we start a computation from the empty database, we see that each database Γ occurring in the computation grows like a tree and Γx corresponds to the path of formulas from the root to point x. The idea of the unlabelled procedure is to consider at each step only a single path of formulas. Notice that if we restrict our attention to the intuitionistic implicational fragment (namely, success, implication, and reduction rule), we obviously have:

$\Gamma \vdash^? x : A$ succeeds iff $\Gamma x \vdash^? x : A$ succeeds.

In case of **LC** , we can switch from one point to another by backtracking or linear restart. We must reformulate these two rules when we take care only of paths of formulas. For backtracking we expect a rule like the following: from $\Gamma x \vdash^? x : q, H$, with q atomic if $(y, r) \in H$ step to

$$\Gamma y \vdash^? y : r, H \cup \{(x, q)\}.$$

For linear restart, we expect the rule: from $\Gamma \vdash^? x : q, H$, if $(y, r) \in H$ step to

(1) $(\Gamma \cup \{y \leq x\})x \vdash^? x : q, H$ and (2) $(\Gamma \cup \{x \leq y\})y \vdash^? y : r, H$

We can observe that the set of formulas in $(\Gamma \cup \{y \leq x\})x$ is the same as the set of formulas in $(\Gamma \cup \{x \leq y\})y$ and is exactly the set of formulas in

$$\Gamma x \cup \Gamma y.$$

These are the formulas of Γ which can be used in the subderivation of (1) and (2). Since given $\Gamma \downarrow x$ and $\Gamma \downarrow y$ we can say which formulas can be used in the subderivation of (1) and (2) without making use of the constraints, we can get rid of the constraints themselves. By this observation, we can hence reformulate the entire procedure, restricting our consideration to databases which are path of formulas, we do not need the constraints (whence the labels) anymore.

3.2.1 Unlabelled procedure for the implicational fragment of LC

We give here a simplified procedure for the implicational fragment of **LC** . The procedure can be extended to the $\rightarrow, \wedge, \bot$-fragment as shown in the previous chapter. This procedure is also inspired by Avron's hypersequent formulation of **LC** [Avron 91a] as we will see at the end of the chapter.

A query has the form

$$\Gamma \vdash^? G, H$$

where Γ is a set of formulas, G is a formula, and H (the history) is a set of pairs of the form (Δ, q), where Δ is a database and q is an atomic query. Here below are the rules

- (success) $\Gamma \vdash^? q, H$ succeeds if $q \in \Gamma$.
- (implication) from $\Gamma \vdash^? A \rightarrow B, H$ we step to

  $\Gamma, A \vdash^? B, H$

- (reduction) from $\Gamma \vdash^? q, H$ if there is a formula $C \in \Gamma$ with $C: A_1 \rightarrow \ldots A_k \rightarrow q$, then we step to

  $\Gamma \vdash^? A_i, H \cup \{(\Gamma, q)\}$ for $i = 1, \ldots, k$

- (backtracking) from $\Gamma \vdash^? q, H$ if $(\Delta, r) \in H$, we step to

  $\Delta \vdash^? r, H \cup \{(\Gamma, q)\}$.

- (linear restart) from $\Gamma \vdash^? q, H$ if $(\Delta, r) \in H$, we step to

$$\Gamma \cup \Delta \vdash^? q, H \quad \text{and} \quad \Gamma \cup \Delta \vdash^? r, H.$$

The linear restart rule allows to combine a previous database with the current database in order to prove both the old goal and the current one. Whereas the backtracking rule does not add any power, it is the linear restart rule which really lead out from intuitionistic logic.

Example 3.2.2 We show that

$$((a \rightarrow b) \rightarrow c) \rightarrow ((b \rightarrow a) \rightarrow c) \rightarrow c$$

succeeds. Here is a derivation:

$\emptyset \vdash^? ((a \to b) \to c) \to ((b \to a) \to c) \to c, \emptyset$
$(a \to b) \to c \vdash^? ((b \to a) \to c) \to c, \emptyset$
$(a \to b) \to c, (b \to a) \to c \vdash^? c, \emptyset$

Let $\Delta = \{(a \to b) \to c, (b \to a) \to c\}$, we go on by reduction wrt. the first formula

$\Delta \vdash^? a \to b, \{(\Delta, c)\}$,
$\Delta \cup \{a\} \vdash^? b, \{(\Delta, c)\}$.

We apply backtracking and we step to:

$\Delta \vdash^? c, \{(\Delta, c), (\Delta \cup \{a\}, b)\}$

We go on by reduction on the second formula of Δ

$\Delta \vdash^? b \to a, \{(\Delta, c), (\Delta \cup \{a\}, b)\}$
$\Delta \cup \{b\} \vdash^? a, \{(\Delta, c), (\Delta \cup \{a\}, b)\}$,

Now we apply linear restart and we step to:

$$\Delta \cup \{a, b\} \vdash^? a, \{(\Delta, c), (\Delta \cup \{a\}, b)\} \quad \text{and} \quad \Delta \cup \{a, b\} \vdash^? b, \{(\Delta, c), (\Delta \cup \{a\}, b)\}.$$

Both of them immediately succeed. ■

We prove next some properties of the computation for **LC** First, we have monotonicity of deduction on both databases and histories.

Proposition 3.2.3 *For every database Γ, Δ, formula A, and histories H, H' we have:*

$Q = \Gamma \vdash^? A, H$ *succeeds implies* $\Gamma, \Delta \vdash^? A, H' \cup H''$ *succeeds,*

where $H'' = \{(\Sigma \cup \Delta, c) : (\Sigma, c) \in H\}$.

Proof. By induction on the height of a successful derivation of the query Q. We only exemplify the case of restart. Suppose Q succeeds by restart, then A is an atom q for some $(\Sigma, r) \in H$, the following queries succeed by a derivation of smaller height:

$\Gamma, \Sigma \vdash^? q, H$ and $\Gamma, \Sigma \vdash^? r, H$.

By induction hypothesis,

$\Gamma, \Sigma, \Delta \vdash^? q, H' \cup H''$ and $\Gamma, \Sigma \cup \Delta \vdash^? r, H' \cup H''$

succeed; but $(\Sigma \cup \Delta, r) \in H''$, whence the query $\Gamma, \Delta \vdash^? A, H' \cup H''$ succeeds. □

Proposition 3.2.4 *Let $A = A_1 \to \ldots \to A_k \to q$, and $B = B_1 \to \ldots \to B_h \to r$, then for any Δ, Γ, H we have*

$$(i)\ \Gamma \vdash^? A, H \cup \{(\Delta \cup \{B_1, \ldots B_h\}, r)\} \text{ succeeds}$$

if and only if

$$(ii)\ \Delta \vdash^? B, H \cup \{(\Gamma \cup \{A_1, \ldots A_k\}, q)\} \text{ succeeds.}$$

Proof. Since the claim is symmetric it suffices to show one half. We start from the query (ii)

$$\Delta \ \vdash^? \ B, H \cup \{(\Gamma \cup \{A_1, \ldots A_k\}, q)\}$$

and we step to

$$\Delta \cup \{B_1, \ldots, B_h\} \ \vdash^? \ r, H \cup \{(\Gamma \cup \{A_1, \ldots A_k\}, q)\}$$

then we apply backtracking and step to

$$(iii)\ \Gamma \cup \{A_1, \ldots A_k\} \ \vdash^? \ q, H \cup \{(\Delta \cup \{B_1, \ldots B_h\}, r), (\Gamma \cup \{A_1, \ldots A_k\}, q)\}.$$

By the implication rule, every successful derivation of (i) contains a successful derivation of

$$\Gamma \cup \{A_1, \ldots A_k\} \ \vdash^? \ q, H \cup \{(\Delta \cup \{B_1, \ldots B_h\}, r)\},$$

thus (iii) succeeds by monotonicity on histories. □

The following property shows that we can restrict the application of both backtracking and restart to the case when we cannot proceed by a successful reduction.

Proposition 3.2.5 *Suppose that $N = \ \Delta \ \vdash^? \ q, H$ succeeds by a derivation $\mathcal{D}$ and there is a clause $C \in \Delta$ such that C is used to reduce q in some descendant of N. Then there is a successful derivation $\mathcal{D}'$ of N such that the first step is the reduction of q with respect to C.*

The following proposition states a sort of "idempotency" (or contraction) property.

Proposition 3.2.6 *For all Δ and H, we have: $\Delta \ \vdash^? \ q, H \cup \{(\Delta, q)\}$ succeeds implies $\Delta \ \vdash^? \ q, H$ succeeds.*

Proof. Let $\mathcal{D}$ be a successful derivation of the query (call it N) in the hypothesis. If the first step in $\mathcal{D}$ is a reduction step or a backtracking step (using $(\Gamma, r) \in H$) then the claim immediately follows: starting from $\Delta \ \vdash^? \ q, H$ we go on the same as in $\mathcal{D}$, (Δ, q) will be added to H after the first step. Suppose that the first step of $\mathcal{D}$ is a restart step through $(\Gamma, r) \in H$; that is from N we step to

$$\Delta, \Gamma \ \vdash^? \ q, H \cup \{(\Delta, q)\} \quad \text{and} \quad \Delta, \Gamma \ \vdash^? \ r, H \cup \{(\Delta, q)\}.$$

Then, from $\Delta \ \vdash^? \ q, H$ we step to $\Gamma \ \vdash^? \ r, , H \cup \{(\Delta, q)\}$, by backtracking, and then by restart, we generate the same queries as above. □

In order to prove completeness we need to prove that cut is admissible. Since the current database may switch with another database a which occurs in the history, we must take into account the occurrences of the the cut-formula in databases in the history, and thus the proof of the cut property is slightly more difficult than in the previous cases. We must carefully formulate the cut property in its right generality in order to make a working inductive proof. After some reflection, we arrive to the following formulation.

Theorem 3.2.7 *Suppose that*

(1) $\Gamma \ \vdash^? \ B, H_1$ *succeeds, and*
(2) $\Delta \ \vdash^? \ A, H_2$ *succeeds.*

Then $\Gamma^* \ \vdash^? \ B, H_1^* \cup H_2$ *succeeds, where* $\Gamma^* = \Gamma$ *or* $\Gamma^* = \Gamma[A/\Delta]$*, and* $H_1^* = \{((\Sigma^*, r) : (\Sigma, r) \in H_1 \ \wedge (\Sigma^* = \Sigma \ \vee \ \Sigma^* = \Sigma[A/\Delta])\}$.

Proof. As usual, we proceed by double induction on $cp(A)$ and the height h of a successful derivation of (1). We only consider the cases which are not a straightforward reformulation of the corresponding ones for intuitionistic logic, namely the cases in which the query (1) is obtained by backtracking, or it is obtained by restart, or $cp(A) > 0$ and query (1) is obtained by reduction with respect to A.

- Suppose that B is an atom q and (1) succeeds by backtracking. Then from (1) we step to $\Sigma \ \vdash^? \ p, H_1 \cup \{(\Gamma, q)\}$, (where $(\Sigma, p) \in H_1$) which succeeds by a shorter derivation. By induction hypothesis we obtain that

  $$(*) \ \Sigma^* \ \vdash^? \ p, H_1^* \cup \{(\Gamma^*, q)\} \cup H_2 \text{ succeeds.}$$

  Since $(\Sigma^*, p) \in H_1^*$, then from $\Gamma^* \ \vdash^? \ q, H_1^* \cup H_2$, we can step to $(*)$ and succeed.

- Suppose that B is an atom q and (1) succeeds by restart. Then from (1) we step to

  $$\Gamma \cup \Sigma \ \vdash^? \ p, H_1 \quad \text{and} \quad \Gamma \cup \Sigma \ \vdash^? \ q, H_1$$

  (where $(\Sigma, p) \in H_1$), and both queries succeed by a shorter derivation. We can apply the induction hypothesis and obtain that

  $$(\Gamma \cup \Sigma)^* \ \vdash^? \ p, H_1^* \cup H_2 \quad \text{and} \quad (\Gamma \cup \Sigma)^* \ \vdash^? \ q, H_1^* \cup H_2 \text{ both succeed.}$$

  Since $(\Gamma \cup \Sigma)^* \subseteq \Gamma^* \cup \Sigma^*$, by monotonicity we obtain that

  $$\Gamma^* \cup \Sigma^* \ \vdash^? \ p, H_1^* \cup H_2 \quad \text{and} \quad \Gamma^* \cup \Sigma^* \ \vdash^? \ q, H_1^* \cup H_2$$

  both succeed. As $(\Sigma^*, p) \in H_1^*$, from $\Gamma^* \ \vdash^? \ q, H_1^* \cup H_2$, we can step to the two queries here above and succeed.

- Suppose that $cp(A) > 0$, B is an atom q and (1) succeeds by reduction with respect to A. Let $A : D_1 \to \ldots \to D_k \to q$ in Γ; from (1) we step to

  $$\Gamma \ \vdash^? \ D_i, H_1 \cup \{(\Gamma, q)\}, \text{for } i = 1, \ldots, k$$

  which succeed by a shorter derivation. By induction hypothesis we obtain that for $i = 1, \ldots, k$ to

  $$(c_i) \ \Gamma^* \ \vdash^? \ D_i, H_1^* \cup \{(\Gamma^*, q)\} \cup H_2$$

  succeed. We have two cases: if $\Gamma^* = \Gamma$, we can still reduce with respect to A and the result easily follows. If $\Gamma^* \neq \Gamma$, then $A \in \Gamma$ and $\Gamma^* = \Gamma[A/\Delta]$. From the hypothesis (2) we have that

  $$(3) \ \Delta \cup \{D_1, \ldots, D_k\} \ \vdash^? \ q, H_2 \text{ succeeds.}$$

  Since $cp(D_i) < cp(A)$ we can repeatedly apply the induction hypothesis, cutting (3) with (c_1) and the result with (c_2), and so on. For instance at the first step we get

  $$\Delta_1 \cup \{D_2, \ldots, D_k\} \cup \Gamma^* \ \vdash^? \ q, H_1^* \cup \{(\Gamma^*, q)\} \cup H_2 \text{ succeeds,}$$

where $\Delta_1 = \Delta - \{D_1\}$ if $D_1 \in \Delta$ and $\Delta_1 = \Delta$ otherwise. Notice that we do not modify H_2 (that is we can let $H_2^* = H_2$). At the end we obtain that

$$\Delta_k \cup \Gamma^* \vdash^? q, H_1^* \cup \{(\Gamma^*, q)\} \cup H_2 \text{ succeeds,}$$

where $\Delta_k \subseteq \Delta$, and hence $\Delta_k \subseteq \Gamma^* = \Gamma[A/\Delta]$. We hence have that $\Gamma^* \vdash^? q, H_1^* \cup \{(\Gamma^*, q)\} \cup H_2$ succeeds, so that by proposition 3.2.6 we finally obtain that $\Gamma^* \vdash^? q, H_1^* \cup H_2$ succeeds.

□

Proposition 3.2.8 *Suppose that* (*i*) $\Gamma, A \to B \vdash^? C, H$ *succeeds and* (*ii*) $\Gamma, B \to A \vdash^? C, H$ *succeeds. Then also* $\Gamma \vdash^? C, \emptyset$ *succeeds.*

Proof. To simplify notation we let

$A = A_1 \to \dots A_n \to q$ and $\Sigma = \{A_1, \dots, A_n\}$,
$B = B_1 \to \dots B_m \to p$ and $\Delta = \{B_1, \dots, B_m\}$,
$C = C_1 \to \dots C_k \to r$ and $\Pi = \{C_1, \dots, C_k\}$.

It is easy to see that the following succeeds:

(1) $\Gamma \vdash^? A \to B, \{(\Gamma \cup \Sigma \cup \{B\}, q)\}$.

To see this, from (1) we apply the implication rule and then restart. From (i) and (1) by cut we get

$$\Gamma \vdash^? C, H \cup \{(\Gamma \cup \Sigma \cup \{B\}, q)\} \text{ succeeds,}$$

and hence also that $\Gamma, \Pi \vdash^? r, H \cup \{(\Gamma \cup \Sigma \cup \{B\}, q)\}$ succeeds; by proposition 3.2.4, we have that

$$\Gamma, \Sigma, B \vdash^? q, H \cup \{(\Gamma \cup \Pi, r)\} \text{ succeeds,}$$

that implies

$$\Gamma \vdash^? B \to A, H \cup \{(\Gamma \cup \Pi, r)\} \text{ succeeds.}$$

From (ii) and the last query, by using cut we obtain

$$\Gamma \vdash^? C, H \cup \{(\Gamma \cup \Pi, r)\} \text{ succeeds,}$$

that implies

$$\Gamma, \Pi \vdash^? r, H \cup \{(\Gamma \cup \Pi, r)\} \text{ succeeds.}$$

By proposition 3.2.6 we have $\Gamma, \Pi \vdash^? r, H;$ succeeds so that we finally have $\Gamma \vdash^? C, H$ succeeds. □

3.2.2 Soundness and Completeness

As we mentioned at the beginning of the section **LC** is complete with respect to the class of *linear* Kripke models i.e. models $M = (W, w, \leq, V)$ where $\leq$ is a *linear order* on W:

$\forall w, w' \in W \; w \leq w' \vee w' \leq w$.

We first show soundness.

Theorem 3.2.9 (Soundness) *Let* $\Gamma = \{B_1, \ldots, B_n\}$, $H = \{(\Delta_1, q_1), \ldots, (\Delta_k, q_k)\}$, *where* $\Delta_i = \{C_{i,1}, \ldots, C_{i,r_i}\}$. *Let* $\bigwedge \Gamma$ *denote the conjunction of formulas of* Γ *(and similarly,* $\bigwedge \Delta_i$ *of* Δ_i*).*
Suppose that $\Gamma \vdash^? A, H$ *succeeds, then*

$$(\bigwedge \Gamma \rightarrow A) \vee (\bigwedge \Delta_1 \rightarrow q_1) \vee \ldots \vee (\bigwedge \Delta_k \rightarrow q_k) \text{ is valid in } \mathbf{LC}\ .$$

Proof. The proof proceeds by induction on the length of derivations. All cases are left to the reader, except for (restart). Let $\Gamma \vdash^? A, H$ succeeds by restart, then A is an atom r, and for some $(\Delta_i, q_i) \in H$, the derivation steps to

$$\Gamma \cup \Delta_i \vdash^? r, H \qquad \Gamma \cup \Delta_i \vdash^? q_i, H.$$

Both queries succeed by derivations of smaller height. Let

$$\bigvee H' = (\bigwedge \Delta_1 \rightarrow q_1) \vee \ldots \vee (\bigwedge \Delta_{i-1} \rightarrow q_{i-1}) \vee (\bigwedge \Delta_{i+1} \rightarrow q_{i+1}) \vee \ldots \vee (\bigwedge \Delta_k \rightarrow q_k).$$

Then, by induction hypothesis, the following are valid in LC:

(1) $(\bigwedge \Gamma \wedge \bigwedge \Delta \rightarrow r) \vee \bigvee H' \vee (\bigwedge \Delta_i \rightarrow q_i)$,
(2) $(\bigwedge \Gamma \wedge \bigwedge \Delta \rightarrow q_i) \vee \bigvee H' \vee (\bigwedge \Delta_i \rightarrow q_i)$.

We must show that also

$$(\bigwedge \Gamma \rightarrow r) \vee \bigvee H' \vee (\bigwedge \Delta_i \rightarrow q_i)$$

is valid in **LC** . Suppose it is not valid, let $M = (W, w, \leq, I)$, be a linear Kripke model which falsifies the above formula. Then, in the initial world w, we have:

(3) $M, w \not\models \bigwedge \Gamma \rightarrow r$,
(4) $M, w \not\models \bigvee H'$,
(5) $M, w \not\models \bigwedge \Delta_i \rightarrow q_i$.

On the other hand, since (1) and (2) are valid in M, we must have

(6) $M, w \models \bigwedge \Gamma \wedge \bigwedge \Delta \rightarrow r$, and
(7) $M, w \models \bigwedge \Gamma \wedge \bigwedge \Delta \rightarrow q_i$.

From (3) and (5), it follows that there are $w', w'' \geq w$ such that

(8) $M, w' \models \bigwedge \Gamma$ and $M, w' \not\models r$,
(9) $M, w' \models \bigwedge \Delta_i$ and $M, w' \not\models q_i$.

By linearity, $w' \leq w''$ or $w'' \leq w'$. Thus, if we let $w^* = max(w', w'')$, we have $w^* = w'$ or $w^* = w$, whence by monotony

$M, w^* \models \bigwedge \Gamma \wedge \bigwedge \Delta$, so that (by (6) and (7))
$M, w^* \models r \wedge q_i$,

which contradicts either (8) or (9). □

We give a semantical proof of the completeness of the procedure by means of a canonical model construction.

Before presenting the completeness proof, we notice that the computation procedure is well defined even in the case the involved databases are infinite. Since a successful derivation is a finite tree, only a finite number of formulas can be involved in it. We thus have an immediate compactness property:

$\Gamma \ \vdash^? \ A, \emptyset$ succeeds if and only if there is a finite subset Γ_0 of Γ such that $\Gamma_0 \ \vdash^? \ A, \emptyset$ succeeds.

We use this property in the completeness proof.

Theorem 3.2.10 *[Completeness] If $\Delta_0 \to G$ is valid in* **LC** *then $\Delta_0 \ \vdash^? \ G, \emptyset$ succeeds.*

Proof. We prove that if $\Delta_0 \ \vdash^? \ G, \emptyset$ does not succeed, then there is a linear model $M = (W, w, \leq, I)$ and an $x \in W$ such that

$$M, x \not\models \Delta_0 \to G.$$

Suppose that $\Delta_0 \ \vdash^? \ G, \emptyset$ does not succeed. Let (A_i, B_i) for $i = 1, 2, \ldots, n, \ldots$, be an enumeration of all pairs of distinct formulas of the language. We define an increasing sequence of databases $\Gamma_0 \subseteq \Gamma_1 \ldots \Gamma_i \ldots$.

Step 0 We let $\Gamma_0 = \Delta_0$.

Step i+1 We consider the pair (A_{i+1}, B_{i+1}) and we proceed as follows:

- if $\Gamma_i, A_{i+1} \to B_{i+1} \ \vdash^? \ G_0, \emptyset$ does not succeed, then we let $\Gamma_{i+1} = \Gamma_i \cup \{A_{i+1} \to B_{i+1}\}$;
- else if $\Gamma_i, B_{i+1} \to A_{i+1} \ \vdash^? \ G_0, \emptyset$ doe not succeed, then we let $\Gamma_{i+1} = \Gamma_i \cup \{B_{i+1} \to A_{i+1}\}$;
- else we let $\Gamma_{i+1} = \Gamma_i$.

We finally let $\Gamma^* = \bigcup_i \Gamma_i$.
Claim 1 for all i, $\Gamma_i \ \vdash^? \ G, \emptyset$ does not succeed. This is easily proved by induction on i.
Claim 2 $\Gamma^* \ \vdash^? \ G, \emptyset$ does not succeed. Suppose it succeeds, then there is a finite subset Γ of Γ^* such that $\Gamma \ \vdash^? \ G, \emptyset$ succeeds. But it must be $\Gamma \subseteq \Gamma_i$ for some i, and we have a contradiction with Claim 1.
Claim 3 For all formulas A, B either $\Gamma^* \cup \{A\} \ \vdash^? \ B, \emptyset$ succeeds, or $\Gamma^* \cup \{B\} \ \vdash^? \ A, \emptyset$ succeeds. This is an easy consequence of the fact either $A \to B \in \Gamma^*$ or $B \to A \in \Gamma^*$, what we prove now. Let $A = A_{i+1}$ and $B = B_{i+1}$, then the pair (A_{i+1}, B_{i+1}) is considered at step $i+1$. If neither $A_{i+1} \to B_{i+1} \in \Gamma_{i+1}$, nor $B_{i+1} \to A_{i+1} \in \Gamma_{i+1}$, then, by definition, both

$\Gamma_i, A_{i+1} \to B_{i+1} \ \vdash^? \ G$ and $\Gamma_i, B_{i+1} \to A_{i+1} \ \vdash^? \ G$ succeeds.

By proposition 3.2.8 we get that $\Gamma_i \ \vdash^? \ G, \emptyset$ succeeds. against Claim 1.

We now define for arbitrary formulas

$A \leq_{\Gamma^*} B$ if and only if $\Gamma^* \cup \{B\} \ \vdash^? \ A, \emptyset$ succeeds;
$A \approx_{\Gamma^*} B$ iff $A \leq_{\Gamma^*} B$ and $B \leq_{\Gamma^*} A$.

We can easily check that:
Claim 4 $\approx_{\Gamma^*}$ is a congruence on formulas.

We consider the quotient W of the language with respect to $\approx_{\Gamma^*}$:

$$W = \{[A]_{\approx_{\Gamma^*}} : A \in \mathcal{L}\}.$$

We still denote by $\leq_{\Gamma^*}$ the relation between equivalence classes: $[A]_{\approx_{\Gamma^*}} \leq_{\Gamma^*} [B]_{\approx_{\Gamma^*}}$ iff $A \leq_{\Gamma^*} B$. Using Claim 3 we can easily check that:
Claim 5 $\leq_{\Gamma^*}$ is a linear order on W.

To simplify notation we omit $\approx_{\Gamma^*}$ from equivalence classes, that is we simply write $[A]$ instead of $[A]_{\approx_{\Gamma^*}}$. We now define a model M by putting

$$M = (W, [A_0], \leq_{\Gamma^*}, V),$$

where where A_0 is, say $p_0 \to p_0$, for some atom p_0, $V : W \to Pow(Var)$ is defined as follows

$p \in V([A]) \Leftrightarrow \Gamma^* \cup \{A\} \vdash^? p, \emptyset$ succeeds.

It is easily seen that the definition is independent from the choice of the representative A, and that V is monotonic, that is:

if $[A] \leq_{\Gamma^*} [B]$ then $I([A]) \subseteq I([B])$.

Claim 6 For all formulas B, and $[A]$, we have

$$M, [A] \models B \Leftrightarrow \Gamma^* \cup \{A\} \vdash^? B, \emptyset \text{ succeeds.}$$

Proof If B is an atom the claim holds by definition. Let $B = C \to D$.

($\Rightarrow$) Suppose $M, [A] \models C \to D$. Either (a) $[A] \leq_{\Gamma^*} [C]$, or (b) $[C] \leq_{\Gamma^*} [A]$. In case (a) we have that $\Gamma^* \cup \{C\} \vdash^? C, \emptyset$ succeeds, so that by induction hypothesis $M, [C] \models C$, and by hypothesis and (a) we have $M, [C] \models D$. By induction hypothesis we obtain $\Gamma^* \cup \{C\} \vdash^? D, \emptyset$ succeeds whence also

$\Gamma^* \cup \{A\} \vdash^? C \to D, \emptyset$ succeeds.

In case (b), by definition we have that $\Gamma^* \cup \{A\} \vdash^? C, \emptyset$ succeeds, and hence by induction hypothesis $M, [A] \models C$. Since $[A] \leq_{\Gamma^*} [A]$, by hypothesis we have $M, [A] \models D$. so that by induction hypothesis we may conclude $\Gamma^* \cup \{A\} \vdash^? D, \emptyset$ succeeds, and therefore $\Gamma^* \cup \{A\} \vdash^? C \to D, \emptyset$ succeeds.
($\Leftarrow$) Suppose that $\Gamma^* \cup \{A\} \vdash^? C \to D, \emptyset$ succeeds, and let (1) $[A] \leq_{\Gamma^*} [E]$ and (2) $M, [E] \models C$. By (1) we have

(3) $\Gamma^* \cup \{E\} \vdash^? A, \emptyset$ succeeds;

by (2) and the induction hypothesis we have that

(4) $\Gamma^* \cup \{E\} \vdash^? C, \emptyset$ succeeds

By hypothesis we have

(5) $\Gamma^* \cup \{A, C\} \vdash^? D, \emptyset$ succeeds

Now cutting (5) with (3), and then with (4) we get $\Gamma^* \cup \{E\} \vdash^? D, \emptyset$ succeeds, so that by induction hypothesis we can conclude $M, [E] \models D$.

To conclude the proof of the theorem, let A be any formula in Γ^*. Since $\Delta_0 \subseteq \Gamma^*$, and, on the other hand, $\Gamma^* \vdash^? G$ does not succeed, by Claim 6, we get

$M, [A] \models B$ for all $B \in \Delta_0$, but
$M, [A] \not\models G$.

Thus $\Delta_0 \to G$ is not true in M. $\square$

3.3 Relation with Avron's hypersequents

Avron has presented a sequent calculus for **LC** [Avron 91a]. His method is based on hypersequents, a generalization of ordinary Gentzen's methods in which the formal objects involved in derivations are disjunctions of ordinary sequents. Hypersequents are denoted by

$$\Gamma_1 \vdash A_1 \mid \ldots \mid \Gamma_n \vdash A_n$$

and can be interpreted as

$$(\Gamma_1 \rightarrow A_1) \vee \ldots \vee (\Gamma_n \rightarrow A_n).$$

(Here we are only concerned with single-conclusion hypersequents, although for specific logics it is required a multi-conclusion version [Avron 87]) Rules for **LC** -hypersequents (as well as for other logics) are divided in two groups: *external* rules which govern whole hypersequents, and *internal* rules which govern formulas within components $\Gamma_i \vdash_i A_i$ of one or more hypersequents. We will not give a complete description of Avron's calculus for **LC** . It suffices to know that some external rules force hypersequents to be interpreted as disjunctions. For example we have *external* weakening, permutation and contraction, the last being:

$$\frac{H \mid \Gamma \vdash A \mid \Gamma \vdash A \mid H'}{H \mid \Gamma \vdash A \mid H'}$$

(here H and H' abbreviate hypersequents). Internal rules are a straightforward generalization of standard rules for intuitionistic logic. In addition there is a specific (external) structural rule, called *communication rule* for **LC** , namely:

$$\frac{H_1 \mid \Gamma_1, \Gamma_2 \vdash A \quad H_2 \mid \Delta_1, \Delta_2 \vdash B}{H_1 \mid H_2 \mid \Gamma_1, \Delta_1 \vdash A \mid \Gamma_2, \Delta_2 \vdash B}$$

We want to show an intuitive mapping between the goal-directed procedure for **LC** and hypersequent calculus. To this purpose, a query

$$\Gamma \ \vdash^? \ A, \{(\Delta_1, q_1), \ldots, (\Delta_n, q_n)\}$$

corresponds to the hypersequent

$$\Gamma \vdash A \mid \Delta_1 \vdash q_1 \mid \ldots \mid \Delta_n \vdash q_n.$$

Given this mapping, we might show that derivations in the goal-directed procedure correspond to a sort of "uniform proofs" according to the terminology of [Miller et al. 91] in Avron's calculus, once we had adapted and extended the notion of uniform proof to this setting. Although we will not develop this correspondence formally here, we point out the main connections. In particular, we want to understand what is the role of backtracking and restart rules in terms of Avron's calculus. The application of backtracking corresponds to an application of *external contraction rule.* Suppose that we have a part of a derivation as shown in fig.3.3. To simplify the matter, we suppose that only implication and reduction rules are applied on the branch from $\Gamma \ \vdash^? \ q, H$ to $\Delta \ \vdash^? \ r, H$. A corresponding hypersequent derivation will contain the branch:

$$\cfrac{\cfrac{\Delta \vdash r \mid \Gamma \vdash q \mid H'}{\cfrac{\vdots}{\Gamma \vdash q \mid \Gamma \vdash q \mid H}}}{\Gamma \vdash q \mid H}$$

$$\begin{array}{c} \Gamma \ \vdash^? \ q, H \\ \vdots \\ \Delta \ \vdash^? \ r, H' \\ \downarrow \\ \Gamma \ \vdash^? \ q, H' \cup \{(\Delta, r)\} \end{array}$$

Figure 3.3:

Notice that the direction of the derivation is inverted.

We come to the restart rule, we can argue that this rule is equivalent to Avron's *communication* rule. To see this, one can easily observe that:

1. The communication rule in Avron's calculus can be restricted to A, B atoms.
2. The communication rule is equivalent to

$$\frac{H \mid \Gamma, \Delta \vdash q \quad H \mid \Gamma, \Delta \vdash r}{H \mid \Gamma \vdash q \mid \Delta \vdash r}$$

This follows from the fact that we have weakening and contraction (both internal and external)

This rule is clearly similar to our restart rule, once that we interpret H as the history. Notice in passim that the above formulation of the communication rule is much more deterministic in backward proof search than Avron's original one: in the above rule once we have fixed the components of the lower sequent (to which apply the rule) the premises are determined, in contrast by Avron's rule, we have an exponential number of choices for the premises.

The relationship with hypersequents may be a source of inspiration for discovering goal-directed formulations of other logics. For example, another well-known intermediate logic (weaker than **LC**) is the so-called logic of weak excluded-middle **LQ** [Hosoi 88] which is obtained by adding to IL the following axiom:

$\neg A \vee \neg\neg A.$

An equivalent implicational axiom is

$((A \to \bot) \to B) \to (((A \to \bot) \to \bot) \to B) \to B.$

The logic **LQ** is complete with respect to the class of bounded Kripke models (or, equivalently the class of Kripke models with a top element). To obtain a goal-directed proof-system for **LQ** , we consider the basic proof system for **I** (for the fragment $(\to, \bot)$, enriched by the history book-keeping mechanism and the backtracking rule of **LC** , then we add the following restart rule:

if $(\Delta, \bot) \in H$, then from $\Gamma \ \vdash^? \ \bot, H$ step to
$\Gamma, \Delta \ \vdash^? \ \bot, H$.

We conjecture that the resulting proof system is sound and complete with respect to **LQ** , we will leave to the interested reader to check it (see [Ciabattoni et al.] for an hypersequent calculus for LQ).

I AM ALMOST SURE THAT IT IS COMPLETE, SHALL WE SAY MORE??

The restart rule for **LC** is connected to the one for classical logic that we have seen in the previous chapter. The connection can be explained in terms of hypersequent calculi by examining Avron's communication rule. We can strengthen the communication rule by discarding one premise:

$$\frac{H \mid \Gamma, \Delta \vdash A}{H \mid \Gamma \vdash B \mid \Delta \vdash A}$$

The calculus obtained by replacing the communication rule by the above one is cut-free and allows to derive for instance $A \vee (A \rightarrow B)$. It is easy to see we have got a calculus for classical logic.

In the goal-directed proof system, the corresponding version of restart would be obtained from the one for **LC** by by discarding one branch:

if $(\Gamma, q) \in H$, then from $\Delta \vdash^? r, H$ step to
$\Gamma, \Delta \vdash^? q, H$.

Let us call this rule last "modified restart". Modified restart rule is equivalent to the restart rule for classical logic. That is to say, the proof system given by the rules for **I** plus modified restart gives classical logic. To see this, we simply observe that in this proof system databases never get smaller along a computation, that is: if $\Gamma \vdash^? q$ precedes $\Delta \vdash^? r$ in a branch of a derivation, then $\Gamma \subseteq \Delta$. Thus, there is no longer need of keeping track of the databases in the history, and the modified restart rule simplifies to

if $q \in H$, then from $\Gamma \vdash^? r, H$ step to $\Delta \vdash^? q, H$.

which is the restart rule for classical logic.

In the literature there are other proposals of calculi for **LC** and other intermediate logics. In [Avellone et al.98] duplication-free tableau and sequent calculi are defined. On the same line, in [Dickhoff 98] terminating calculi for theorems and non-theorems of propositional **LC** are presented. These calculi do not go bejond the format of standard Gentzen calculi and contain *global* rules (similar to the standard calculi for modal logic [Gore 99]) which may act on several formulas at once [1].

A rather different calculus for **LC** is presented in [Baaz and Fermüller 99], where **LC** is considered a prominent example of *projective logic*.

[1]The characteristic rule for **LC** has this form:

$$\frac{\Gamma, A_1 \vdash B_1, \Delta_1 \ldots \Gamma, A_n \vdash B_n, \Delta_n}{\Gamma \vdash \Delta}$$

where $A_1 \rightarrow B_1, \ldots, A_n \rightarrow B_n$ are all the implicational formulas of Δ and Δ_i is $\{A_1 \rightarrow B_1, \ldots, A_{i-1} \rightarrow B_{i-1}, A_{i+1} \rightarrow B_{i+1}, \ldots, A_n \rightarrow B_n\}$. In [Dickhoff 98] the application of this rule is restricted to so-called *irreducible sequents.*

Chapter 4

Modal Logics of Strict Implication

The purpose of this chapter is to extend the goal directed proof methods to strict implication logics. We consider this as a first step in order to extend the goal-directed paradigm to the realm of modal logics. Strict-implication, denoted by $A \Rightarrow B$ is read as "necessarily A implies B". The notion of necessity (and the dual notion of possibility) are the subject of modal logics. Strict implication can be regarded as a derived notion: $A \Rightarrow B = \Box(A \rightarrow B)$, where $\rightarrow$ denotes material implication and $\Box$ denotes modal necessity. However, strict implication can also be considered as a primitive notion, and has been considered as such already at the beginning of the century in many discussions about the paradoxes of material implication.

PLEASE ADD COMMENTS

The extension of the goal-directed approach to strict implication and modal logics relies upon the *possible-world* semantics of modal logics which is due to Kripke. As we have already done in the case of intuitionistic and intermediate logics, we regard a database as a set of labelled formulas $x_i : A_i$ equipped by a relation α giving connections between labels. The labels represent worlds, states, positions. Thus, $x_i : A_i$ means that A_i holds at x_i. The goal of a query is always asked with respect to a position/world. The form of databases and goals determine the notion of consequence relation

$$\{x_1 : A_1, \ldots, x_n : A_n\}, \alpha \vdash x : A$$

whose intended meaning is that if A_i holds at x_i (for $i = 1, \ldots, n$) and the x_i are connected as α prescribes, then A must hold at x.

For different logics α will be required to satisfy different properties such as reflexivity transitivity etc, depending on the property of the accessibility relation of the system under consideration.

In most of the chapter we will be concerned with implicational modal logics whose language $\mathcal{L}(\Rightarrow)$ contains all formulas built out from a denumerable set Var of propositional variables by applying the strict implication connective, that is, if $p \in Var$ then p is a formula of $\mathcal{L}(\Rightarrow)$, and if A and B are formulas of $\mathcal{L}(\Rightarrow)$, then so is $A \Rightarrow B$. Let us fix an atom p_0, we define the constant $true \equiv p_0 \Rightarrow p_0$ and $\Box A \equiv true \Rightarrow A$.

Semantics
We review the standard Kripke semantics for $\mathcal{L}(\Rightarrow)$.

A Kripke structure M for $\mathcal{L}(\Rightarrow)$ is a triple (W, R, V), where W is a non empty set (whose elements are called possible worlds), R is a binary relation on W, and V is a mapping from W to sets of propositional variables of $\mathcal{L}$. Truth conditions for formulas (of $\mathcal{L}(\Rightarrow)$) are defined as follows:

Name	Reflexivity	Transitivity	Symmetry	Euclidean	Finite chains
K					
T	*				
K4		*			
S4	*	*			
K5				*	
K45		*		*	
KB			*		
KTB	*		*		
S5	*	*	*	*	
G		*			*

Figure 4.1:

- $M, w \models p$ iff $p \in V(w)$;
- $M, w \models A \Rightarrow B$ iff for all w' such that wRw' and $M, w' \models A$, it holds $M, w' \models B$.

We say that a formula A is *valid* in a structure M, denoted by $M \models A$, if $\forall w \in W,\ M, w \models A$. We say that a formula A is *valid* with respect to a given class of structures $\mathcal{K}$, iff it is valid in every structure $M \in \mathcal{K}$, we sometime use the notation $\models_{\mathcal{K}} A$. Let us fix a class of structures $\mathcal{K}$, given two formulas A and B, we can define the *consequence relation* $A \models_{\mathcal{K}} B$ as

$$\forall M \in \mathcal{K}\ \forall w \in W \quad \text{if} \quad M, w \models A \text{ then } M, w \models B.$$

Different modal logics are obtained by considering classes of structures whose relation R satisfy some specific properties. We will take into consideration strict implication $\Rightarrow$ as defined in systems **K**, **T**, **K4**, **S4**, **K5**, **K45**, **KB**, **KBT**, **S5** and **G**[1].

Properties of accessibility relation R in Kripke frames, corresponding to these systems are shown in Fig.5.2.1

Hilbert-style axiomatizations of fragments of strict implication has been given in [Prior 61, Prior 63, Meredith and Prior 64, Corsi 87]. Letting **S** be one of the modal systems above, we use the notation $\models_{\mathbf{S}} A$ (and $A \models_{\mathbf{S}} B$) to denote validity in (and the consequence relation determined by) the class of structures corresponding to S.

4.1 Proof Systems

In this section we present proof methods for all modal systems mentioned above with the exception of Gödel Logic G.

Definition 4.1.1 Let us fix a denumerable alphabet $\mathcal{A} = \{x_1, \ldots, x_i, \ldots\}$ of labels. A *database* is a finite graph of formulas labelled by $\mathcal{A}$. We denote a database as a pair (Δ, α), where Δ is a finite set of labelled formulas $\Delta = \{x_1 : A_1, \ldots, x_n : A_n\}$ and $\alpha = \{(x_1, x'_1), \ldots, (x_n, x'_n)\}$ is a set of links; Denoting by $Lab(E)$ the set of labels $x \in \mathcal{A}$ occurring in E, we assume that (i) $Lab(\Delta) = Lab(\alpha)$, and (ii) if $x : A \in \Delta, x : B \in \Delta$, then $A = B$ [2].

[1]We do not consider systems containing $D : \Box A \to \Diamond A$, which corresponds to *seriality* in Kripke frames. The reason is that seriality cannot be expressed in the language of pure strict implication; moreover, it cannot be expressed in any modal language, unless $\neg$ or $\Diamond$ is allowed.

[2]We will drop this condition in section 4.6 when we extend the language by allowing conjunction.

A *trivial database* has the form $(\{x_0 : A\}, \emptyset)$.

The *expansion* of a database (Γ, α) by $y : C$ at x, with $x \in Lab(\Gamma)$, $y \notin Lab(\Gamma)$ is defined as follows:

$$(\Gamma, \alpha) \otimes_x (y : C) = (\Delta \cup \{y : C\}, \alpha \cup \{(x, y)\}).$$

■

Definition 4.1.2 A query Q is an expression of the form:

$$Q = \ (\Delta, \alpha) \ \vdash^? \ x : G, H$$

where (Δ, α) is a database, $x \in Lab(\Delta)$, G is a formula, and H, called *the history*, is a set of pairs

$$H = \{(x_1, q_1), \ldots, (x_m, q_m)\},$$

where $x_i \in Lab(\Delta)$, and q_i are atoms. We will often omit the parentheses around the two components of a database and write $Q = \Delta, \alpha \ \vdash^? \ x : G, H$. A query from a trivial database $\{x_0 : A\}$ will be written simply as:

$$x_0 : A \ \vdash^? \ x_0 : B, H,$$

and if $A = true$, we sometime write just $\vdash^? \ x_0 : A, H$. ■

Definition 4.1.3 Let α be a set of links, we introduce a family of relation symbols $A^{\mathbf{S}}_{\alpha}(x, y)$, where $x, y \in Lab(\alpha)$. We consider the following conditions:

(**K**) $(x, y) \in \alpha \ \Rightarrow \ A^{\mathbf{S}}_{\alpha}(x, y)$,
(**T**) $x = y \ \Rightarrow \ A^{\mathbf{S}}_{\alpha}(x, y)$,
(**4**) $\exists z(A^{\mathbf{S}}_{\alpha}(x, z) \wedge A^{\mathbf{S}}_{\alpha}(z, y)) \ \Rightarrow \ A^{\mathbf{S}}_{\alpha}(x, y)$,
(**5**) $\exists z(A^{\mathbf{S}}_{\alpha}(z, x) \wedge A^{\mathbf{S}}_{\alpha}(z, y)) \ \Rightarrow \ A^{\mathbf{S}}_{\alpha}(x, y)$,
(**B**) $A^{\mathbf{S}}_{\alpha}(x, y) \ \Rightarrow \ A^{\mathbf{S}}_{\alpha}(y, x)$.

For $\mathbf{K} \in \mathbf{S} \subseteq \{\mathbf{K}, \mathbf{T}, \mathbf{4}, \mathbf{5}, \mathbf{B}\}$, we let $A^{\mathbf{S}}$ be the least relation satisfying all conditions in S. Thus, for instance, $A^{\mathbf{K45}}$ is the least relation such that:

$$\begin{array}{rcl} A^{\mathbf{K45}}_{\alpha}(x, y) & \Leftrightarrow & (x, y) \in \alpha \vee \\ & & \vee \ \exists z(A^{K45}_{\alpha}(x, z) \wedge A^{\mathbf{K45}}_{\alpha}(z, y)) \vee \\ & & \vee \ \exists z(A^{K45}_{\alpha}(z, x) \wedge A^{\mathbf{K45}}_{\alpha}(z, y)) \end{array}$$

We will use the standard abbreviations (i.e. $A^{\mathbf{S5}} = A^{\mathbf{KT5}} = A^{\mathbf{KT45}}$).

■

Definition 4.1.4 [Deduction Rules] For each modal system $\mathbf{S}$, the corresponding proof system, denoted by $\mathrm{P}(\mathbf{S})$, comprises the following rules, parametrized to predicates $A^{\mathbf{S}}$.

- (success) $\Delta, \alpha \ \vdash^? \ x : q, H$ immediately succeeds if q is an atom and $x : q \in \Delta$.
- (implication) from $\Delta, \alpha \ \vdash^? \ x : A \Rightarrow B, H$, step to

  $$(\Delta, \alpha) \otimes_x (y : A) \ \vdash^? \ y : B, H,$$

  where $y \notin Lab(\Delta)$.

- (reduction) if $y : C \in \Delta$, with $C = B_1 \Rightarrow B_2 \Rightarrow \ldots \Rightarrow B_k \Rightarrow q$, with q atomic, then from

  $$\Delta, \alpha \vdash^? x : q, H$$

  step to

  $$\Delta, \alpha \vdash^? u_1 : B_1, H \cup \{(x,q)\}, \quad \ldots, \quad \Delta, \alpha \vdash^? u_k : B_k, H \cup \{(x,q)\},$$

  for some $u_0, \ldots, u_k \in Lab(\alpha)$, with $u_0 = y$, $u_k = x$, such that

  for $i = 0, \ldots, k-1$, $A^{\mathbf{S}}_{\alpha}(u_i, u_{i+1})$ holds.

- (restart) if $(y, r) \in H$, then, from $\Delta, \alpha \vdash^? x : q, H$, with q atomic, step to

  $$\Delta, \alpha \vdash^? y : r, H \cup \{(x,q)\}$$

■

Since most of the results which follows do not depend on the specific properties of the predicates $A^{\mathbf{S}}$, involved in the definition of a proof system $\mathrm{P}(\mathbf{S})$, we will often omit the reference to $\mathrm{P}(\mathbf{S})$. The following proposition states easy properties of the deduction procedure.

Proposition 4.1.5 • *(Identity) If $x : A \in \Gamma$, then $\Gamma, \alpha \vdash^? x : A, H$ succeeds.*

- *(Monotony) If $Q = \Gamma, \alpha \vdash^? x : C, H$ succeeds and $\Gamma \subseteq \Delta$, $\alpha \subseteq \beta$, $H \subseteq H'$, then also*

  $Q' = \Delta, \beta \vdash^? x : C, H'$ succeeds.

  Moreover, any derivation of Q can be turned uniformly into a derivation of Q' by replacing Γ with Δ, α with β, and H with H'.

- *(Increasingness) Let $\mathcal{D}$ be any derivation of a given query; if $Q_1 = \Gamma_1, \alpha_1 \vdash^? x_1 : A_1, H_1$ and $Q_2 = \Gamma_2, \alpha_2 \vdash^? x_2 : A_2, H_2$ are two queries in $\mathcal{D}$, such that Q_2 is a descendant of Q_1, then $\Gamma_1 \subseteq \Gamma_2$, $\alpha_1 \subseteq \alpha_2$, $H_1 \subseteq H_2$.*

Restricted restart

Similarly to the case of classical logic, in any deduction of a query Q of the form $\Delta, \alpha \vdash^? x : G, \emptyset$, the *restart rule* can be restricted to the choice of the pair (y, r), such that r is the uppermost atomic goal occurred in the deduction and y is the label associated to r (that is, the query in which r appears contains $\ldots \vdash^? y : r$). Hence, if the initial query is $Q = \Delta, \alpha \vdash^? x : G, \emptyset$ and G is an atom q, such a pair is (x, q), if G has the form $B_1 \Rightarrow \ldots \Rightarrow B_k \Rightarrow r$, then the first pair is obtained by repeatedly applying the implication rule until we reach the query $\ldots \vdash^? x_k : r$, with $x_k \notin Lab(\Delta)$. With this restriction, we do not need to keep track of the history anymore, but only of the first pair. An equivalent formulation is to allow restart from the initial goal (and its relative label) even if it is implicational, but the re-evaluation of an implication causes a redundant increase of the database, that is why we have preferred the above formulation.

Proposition 4.1.6 *If $\Delta, \alpha \vdash^? x : G, \emptyset$ succeeds then it succeeds by a derivation in which every application of restart is* restricted restart.

$$\begin{tikzcd}
& (1)\ \vdash^{?}\ x_0 : ((p \Rightarrow p) \Rightarrow a \Rightarrow b) \Rightarrow (b \Rightarrow c) \Rightarrow a \Rightarrow c \arrow[d, dash] & \\
& (2)\ x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b,\ \alpha\ \vdash^{?}\ x_1 : (b \Rightarrow c) \Rightarrow a \Rightarrow c \arrow[d, dash] & \\
& (3)\ x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b, x_2 : b \Rightarrow c,\ \alpha\ \vdash^{?}\ x_2 : a \Rightarrow c \arrow[d, dash] & \\
& (4)\ x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b, x_2 : b \Rightarrow c, x_3 : a,\ \alpha\ \vdash^{?}\ x_3 : c \arrow[d, dash] & \\
& (5)\ x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b, x_2 : b \Rightarrow c, x_3 : a,\ \alpha\ \vdash^{?}\ x_3 : b,\ (\mathbf{x_3}, \mathbf{c}) \arrow[dl, dash] \arrow[dr, dash] & \\
(6)\ x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b, x_2 : b \Rightarrow c, x_3 : a,\ \alpha\ \vdash^{?}\ x_2 : p \Rightarrow p,\ (\mathbf{x_3}, \mathbf{c}) \arrow[d, dash] & & (8)\ x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b, x_2 : b \Rightarrow c, x_3 : a,\ \alpha\ \vdash^{?}\ x_3 : a,\ (\mathbf{x_3}, \mathbf{c}) \\
(7)\ x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b, x_2 : b \Rightarrow c, x_3 : a, x_4 : p, \alpha \cup \{(x_2, x_4)\}\ \vdash^{?}\ x_4 : p,\ (x_3, c) & &
\end{tikzcd}$$

Figure 4.2:

Proof. It suffices to show the following fact: whenever in a successful derivation we restart from a pair (x, p) coming from a query Q preceding the current one, we still obtain a successful derivation if we restart from any pair (y, q) coming from a query Q' preceding Q. This fact implies that, if the initial query succeeds, then there is a successful derivation in which every restart application makes use only of the first pair. The proof of this fact is essentially identical to the one of the $\Rightarrow$ half of proposition 2.3.17. We omit the details. □

Example 4.1.7 Here below we show a derivation of

$$((p \Rightarrow p) \Rightarrow a \Rightarrow b) \Rightarrow (b \Rightarrow c) \Rightarrow a \Rightarrow c.$$

in P(**K**). By proposition 4.1.6, we only record the first pair for restart, which however it is not used in the following derivation. Here is an explanation of the steps: in step (2), $\alpha = \{(x_0, x_1)\}$, in step (3) $\alpha = \{(x_0, x_1), (x_1, x_2)\}$, in step (4) $\alpha = \{(x_0, x_1), (x_1, x_2), (x_2, x_3)\}$, since $A^{\mathbf{K}}_{\alpha}(x_2, x_3)$, by reduction wrt. $x_2 : b \Rightarrow c$ we get (5), since $A^{\mathbf{K}}_{\alpha}(x_1, x_2)$ and $A^{\mathbf{K}}_{\alpha}(x_2, x_3)$, by reduction wrt. $x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b$ we get(6) and (7) which immediately succeeds, since $x_3 : a \in \Delta$; from (6) we step to (8) which immediately succeeds.

■

Example 4.1.8 We show the a derivation of

$$((((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p) \Rightarrow p$$

in P(**KBT**), we use restricted restart according to proposition 4.1.6. In step (2), $\alpha = \{(x_0, x_1)\}$. Step (3) is obtained by reduction wrt. $x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p$, as $A^{\mathbf{KBT}}_{\alpha}(x_1, x_1)$. In step (4) $\alpha = \{(x_0, x_1), (x_1, x_2)\}$; step (5) is obtained by restart; step (6) by reduction wrt. $x_2 : (a \Rightarrow a) \Rightarrow p$, as $A^{\mathbf{KBT}}_{\alpha}(x_2, x_1)$; in step (7) $\alpha = \{(x_0, x_1), (x_1, x_2), (x_1, x_3)\}$ and the query immediately succeeds.

■

$$\begin{array}{c}
(1)\ \ \vdash^{?}\ x_0 : ((((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p) \Rightarrow p \\
| \\
(2)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, \alpha\ \vdash^{?}\ x_1 : p \\
| \\
(3)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p,\ \alpha\ \vdash^{?}\ x_1 : ((a \Rightarrow a) \Rightarrow p) \Rightarrow q,\ (\mathbf{x_1}, \mathbf{p}) \\
| \\
(4)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, x_2 : (a \Rightarrow a) \Rightarrow p,\ \alpha\ \vdash^{?}\ x_2 : q,\ (\mathbf{x_1}, \mathbf{p}) \\
| \\
(5)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, x_2 : (a \Rightarrow a) \Rightarrow p,\ \alpha\ \vdash^{?}\ x_1 : p,\ (\mathbf{x_1}, \mathbf{p}) \\
| \\
(6)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, x_2 : (a \Rightarrow a) \Rightarrow p,\ \alpha\ \vdash^{?}\ x_1 : a \Rightarrow a,\ (\mathbf{x_1}, \mathbf{p}) \\
| \\
(7)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, x_2 : (a \Rightarrow a) \Rightarrow p, x_3 : a, \alpha\ \vdash^{?}\ x_3 : a,\ (\mathbf{x_1}, \mathbf{p})
\end{array}$$

Figure 4.3:

4.2 Admissibility of Cut

In this section we prove the admissibility of the cut rule. The cut rule states the following: let $x : A \in \Gamma$, then if (1) $\Gamma \vdash y : B$ and (2) $\Delta \vdash z : A$ succeed, we can "replace" $x : A$ by Δ in Γ and get a successful query from (1). There are two points to clarify. First we need to define the involved notion of substitution, and this we do in the next definition. Furthermore the proof systems P($\mathbf{S}$) depend uniformly on predicate $A^{\mathbf{S}}$, we expect that the admissibility of cut on the properties of predicate $A^{\mathbf{S}}$. It will turn out that the admissibility of cut (proved in theorem 4.2.3) will hold for every proof system P($\mathbf{S}$) , such that $A^{\mathbf{S}}$ satisfy the following conditions:

- (i) $A^{\mathbf{S}}$ is closed under substitution,
- (ii) $A^{\mathbf{S}}_{\alpha}(x, y)$ implies $A^{\mathbf{S}}_{\alpha\cup\beta}(x, y)$;
- (iii) $A^{\mathbf{S}}_{\alpha}(u, v)$ implies $\forall x\ y\ (A^{\mathbf{S}}_{\alpha\cup\{(u,v)\}}(x, y) \leftrightarrow A^{\mathbf{S}}_{\alpha}(x, y))$.

Definition 4.2.1 [Substitution] Given a (set of) labelled expression(s) E, we denote by $E[u/v]$ the (set of) expression(s) obtained by replacing all occurrences of a label u by the label v.

We say that two databases (Γ, α), (Δ, β) are *compatible for substitution* [3], if

for every $x \in Lab(\Gamma) \cap Lab(\Delta)$, for all formulas C, $x : C \in \Gamma \Leftrightarrow x : C \in \Delta$;

If (Γ, α) and (Δ, β) are compatible for substitution, $x : A \in \Gamma$, and $y \in Lab(\Delta)$, we denote by

$$(\Gamma, \alpha)[x : A/\Delta, \beta, y] = (\Gamma - \{x : A\} \cup \Delta, \alpha[x/y] \cup \beta).$$

the database which results by replacing $x : A$ in (Γ, α) by (Δ, β) at point y.

We say that a predicate $A^{\mathbf{S}}$ is *closed under substitution* if,

[3]We will drop this condition in section 8, when we add conjunction.

whenever $A^{\mathbf{S}}_{\alpha}(x, y)$ holds, then also $A^{\mathbf{S}}_{\alpha[u/v]}(x[u/v], y[u/v])$ holds.

■

Proposition 4.2.2 • *(a) If $A^{\mathbf{S}}$ satisfies condition (i) above and $\Gamma, \alpha \vdash^{?} x : C, H$ succeeds, then also*

$$\Gamma[u/v], \alpha[u/v] \vdash^{?} x[u/v] : C, H[u/v] \textit{ succeeds.}$$

- *(b) If $A^{\mathbf{S}}$ satisfies condition (iii) above, then $A^{\mathbf{S}}_{\alpha}(x, y)$ and $\Gamma, \alpha \cup \{(x, y)\} \vdash^{?} u : G, H$ succeeds, implies $\Gamma, \alpha \vdash^{?} u : G, H$ succeeds.*

Proof. By induction on derivation lenght. □

Theorem 4.2.3 (Admissibility of cut) *Let predicate $A^{\mathbf{S}}$ satisfy the conditions (i),(ii),(iii) above. If the following queries succeed in a proof system P($\mathbf{S}$) :*

(1) $\Gamma[x : A] \vdash^{?} u : B, H_1$
(2) $\Delta, \beta \vdash^{?} y : A, H_2$.

and (Γ, α) *and* (Δ, β) *are compatible for substitution, then also*

(3) $(\Gamma, \alpha)[x : A/\Delta, \beta, y] \vdash^{?} u[x/y] : B, H_1[x/y] \cup H_2$ *succeeds in P($\mathbf{S}$) .*

Proof. As usual, we proceed by double induction on pairs (h, c), where h is the height of a derivation of (1) and $c = cp(A)$. We only show the most difficult case, namely when $h, c > 0$ and the first step in a derivation of (1) is by reduction with respect to $x : A$. In such a case, letting $A = D_1 \Rightarrow \ldots \Rightarrow D_k \Rightarrow q$; then from (1) we step to

$$\Delta[x : A], \alpha \vdash^{?} u_i : D_i, H_1 \cup \{(u, q)\},$$

for some $u_0, \ldots, u_k \in Lab(\alpha)$, with $u_0 = x$, $u_k = u$, such that

(*) for $i = 0, \ldots, k-1$, $A^{\mathbf{S}}_{\alpha}(u_i, u_{i+1})$ holds.

By induction hypothesis, we get for $i = 0, \ldots, k-1$,

(Q_i) $(\Delta[x : A], \alpha)[x : A/\Delta, \beta, y]$
$\vdash^{?} u_i[x/y] : D_i, (H_1[x/y] \cup \{(u, q)[x/y]\} \cup H_2$ succeeds.

By (2), we get

$(\Delta, \beta) \otimes_y (z_1 : D_1) \otimes_{z_1} \ldots \otimes_{z_{k-1}} (z_k : D_k) \vdash^{?} z_k : q, H_2$ succeeds,

where we can assume $z_1, \ldots, z_k \notin Lab(\Gamma) \cup Lab(\Delta)$ and the z_i are all distinct. That is to say,

(Q') $\Delta \cup \{z_1 : D, \ldots, z_k : D_k\}, \beta \cup \gamma \vdash^{?} z_k : q, H_2$ succeeds,

with $\gamma = \{(y, z_1), \ldots, (z_{k-1}, z_k)\}$. Notice that (i), $cp(D_i) < cp(A)$, for $i = 1, \ldots, k$ and (ii) (Q') and (Q_1) are compatible for substitution, whence we can apply the induction hypothesis and get that the following query succeeds:

$(\Delta \cup \{z_1 : D, \ldots, z_k : D_k\}, \beta \cup \gamma)[z_1/(\Gamma, \alpha)[x/(\Delta, \beta, y)]$
$\vdash^{?} z_k : q, H_2[z_1/u_1[x/y]] \cup H_1[x/y] \cup H_2$.

By the hypothesis on z_i, we also have $z_i \notin Lab(H_2)$, thus by definition of substitution, we have that:

(Q'_1) $(\Gamma - \{x : A\}) \cup \Delta \cup \{z_2 : D_2, \ldots, z_k : D_k\}, \alpha[x/y] \cup \beta \cup \gamma[z_1/u_1[x/y]]$
$\vdash^? \; z_k : q, H_1[x/y] \cup H_2$ succeeds.

The compatibility constraint is satisfied by Q'_1 and Q_2, and since $cp(D_2) < cp(A)$ we can apply the induction hypothesis again. By repeating this argument up to D_k, we finally get:

$(\Gamma - \{x : A\}) \cup \Delta, \alpha[x/y] \cup \beta \cup \gamma[z_1/u_1[x/y], \ldots, z_k/u_k[x/y]]$
$\vdash^? \; u_k[x/y] : q, H_1[x/y] \cup H_2$ succeeds,

so that by definition of γ, since $u_0 = x$ and $u_k = u$,

$(Q")$ $(\Gamma - \{x : A\}) \cup \Delta, \alpha[x/y] \cup \beta \cup \{(y, u_1[x/y]), \ldots,$
$(u_{k-1}[x/y], u[x/y])\} \; \vdash^? \; u[x/y] : q, H_1[x/y] \cup H_2$ succeeds.

By (*) and conditions (i) and (ii) on $A^{\mathbf{S}}$, we get:

$A^{\mathbf{S}}_{\alpha[x/y]\cup\beta}(y, u_1[x/y])$,
$\vdots$
$A^{\mathbf{S}}_{\alpha[x/y]\cup\beta}(u_{k-1}[x/y], u_1[x/y])$.

Hence, by repeatedly applying proposition 4.2.2(b) to query (Q"), we get that

$$(\Gamma - \{x : A\}) \cup \Delta, \alpha[x/y] \cup \beta \; \vdash^? \; u[x/y] : q, H_1[x/y] \cup H_2,$$

that is (3), succeeds. □

Corollary 4.2.4 *At the same conditions as above, if $x : A \; \vdash^? \; x : B$ succeeds and $x : B \; \vdash^? \; x : C$ succeeds then also $x : A \; \vdash^? \; x : C$ succeeds.*

Corollary 4.2.5 *If $\mathbf{K} \in \mathbf{S} \subseteq \{\mathbf{K}, \mathbf{4}, \mathbf{5}, \mathbf{B}, \mathbf{T}\}$, then in the proof system P($\mathbf{S}$) cut is admissible.*

Proof. One can easily check that the predicate $A^{\mathbf{S}}$ satisfies the conditions of the previous theorem. □

4.3 Soundness and Completeness

We need to give a semantic meaning to queries. We do this by introducing a suitable notion of realization, similarly to what we did in the cases of intuitionistic and intermediate logic (see definition 2.5.6, 3.1.4)

Definition 4.3.1 [Realization] Let $A^{\mathbf{S}}$ be an accessibility predicate, given a database (Γ, α) and a Kripke model $M = (W, R, V)$, a mapping $f : Lab(\Gamma) \rightarrow W$ is called a *realization* of (Γ, α) in M with respect to $A^{\mathbf{S}}$, if the following hold

- (a) $A^{\mathbf{S}}_{\alpha}(x, y)$ implies $f(x)Rf(y)$;
- (b) if $x : A \; \in \Gamma$, then $M, f(x) \models A$.

We say that a query $Q = \Gamma \;, \alpha \; \vdash^? \; x : G, H$ is *valid* if for every $\mathbf{S}$-model M and every realization f of (Γ, α), we have either $M, f(x) \models G$, or for some $(y, r) \in H$, $M, f(y) \models r$. ■

Theorem 4.3.2 (Soundness) *Let $Q = \Gamma, \alpha \vdash^? x : G, H$ succeeds in the proof system P(**S**) , then it is valid.*

Proof. Let $M = (W, R, V)$ be an **S**-model and f be a realization of (Γ, α) in M, we proceed by induction on the height h of a successful derivation of Q. For the base case, we have that Q immediately succeeds, G is atomic, and $x : q \in \Delta$; by hypothesis $M, f(x) \models q$.

In the induction step, we have several cases. Let the first step in the derivation be obtained by *implication*, then $G = A \Rightarrow B$. Suppose by way of contradiction that $M, f(x) \not\models A \Rightarrow B$ and for all $(y, r) \in H$, $M, f(y) \not\models r$. We have that for some $w \in W$, such that $f(x)Rw$, $M, w \models A$, but $M, w \not\models B$. From Q we step to

$$Q' = (\Gamma, \alpha) \otimes_x (u : A) \vdash^? u : B, H, \text{ with } u \notin Lab(\Delta),$$

which succeeds by a derivation of smaller height. Let $f'(z) = f(z)$, for $z \neq u$ and $f'(u) = w$. Then, f' is a realization of $(\Gamma, \alpha) \otimes_x (u : A)$, and by induction hypothesis, either $M, f'(u) \models B$, whence $M, w \models B$, or for some $(y, r) \in H$, $M, f'(y) \models r$, whence $M, f(y) \models r$; in both cases we have a contradiction.

Let the first step in the derivation be obtained by *reduction*, then G is an atom q, there is $z : C \in \Delta$, with $C = B_1 \Rightarrow \ldots \Rightarrow B_k \Rightarrow q$, and from Q we step to

$$\Delta, \alpha \vdash^? u_1 : B_1, H \cup \{(x, q)\}, \quad \ldots, \quad \Delta, \alpha \vdash^? u_k : B_k, H \cup \{(x, q)\},$$

for some $u_0, \ldots, u_k \in Lab(\alpha)$, with $u_0 = z$, $u_k = x$, such that

for $i = 0, \ldots, k-1$, $A^{\mathbf{S}}_{\alpha}(u_i, u_{i+1})$ holds.

By hypothesis, we have

(1) $M, f(z) \models B_1 \Rightarrow \ldots \Rightarrow B_k \Rightarrow q$, and
(2) $f(u_i)Rf(u_{i+1})$, for $i = 0, \ldots, k$.

By induction hypothesis, either (a) for some $(y, r) \in H$, $M, f(y) \models r$, or (b) for $i = 1, \ldots, k$, $M, f(u_i) \models B_i$. In case (a) we are done. Suppose (b) holds. From $u_0 = z$, (1) and (2), we get

$M, f(u_i) \models B_{i+1} \Rightarrow \ldots \Rightarrow B_k \Rightarrow q$, for $i = 1, \ldots, k-1$,

and finally $M, f(u_k) \models q$, that is $M, f(x) \models q$.

If the first step in the derivation be obtained by restart, then the claim immediately follows by the induction hypothesis. □

Corollary 4.3.3 *If $x_0 : A \vdash^? x_0 : B, \emptyset$ succeeds in P(**S**) , then $A \models_{\mathbf{S}} B$ holds. In particular, if $\vdash^? x_0 : A, \emptyset$ succeeds in P(**S**) , then A is valid in S.*

Theorem 4.3.4 (Completeness) *Given a query $Q = \Gamma, \alpha \vdash^? x : A, H$ If Q is valid then Q succeeds in the proof system P(**S**) .*

By contrapposition, we prove that if $Q = \Gamma, \alpha \vdash^? x : A, H$ does not succeeds in one proof system P(**S**) , then there is an **S**-model M and a realization f of (Γ, α), such that $M, f(x) \not\models A$ and for any $(y, r) \in H$, $M, f(y) \not\models r$. The proof is very similar to the one of theorem 2.5.10 for the completeness of the procedure of intuitionistic logic with disjunction.

As usual, we construct an **S**-model by extending the database, through the evaluation of all possible formulas at every world (each represented by one label) of the database. Since such evaluation may lead, for implication formulas, to create new worlds, we must carry on the evaluation process on these new worlds. For this reason we consider in the construction an enumeration of pairs (x_i, A_i), where x_i is a label and A_i is a formula.

Assume $\Gamma, \alpha \ \vdash^? \ x : A, H$ fail in P(**S**) . We let $\mathcal{A}$ be a denumerable alphabet of labels and $\mathcal{L}$ be the underlying propositional language. Let (x_i, A_i), for $i \in \omega$ be an enumeration of pairs of $\mathcal{A} \times \mathcal{L}$, starting with the pair (x, A) and containing infinitely many repetitions, that is

$(x_0, A_0) = (x, A)$,
$\forall y \in \mathcal{A}, \forall F \in \mathcal{L}, \forall n \ \exists m > n \ (y, F) = (x_m, A_m)$.

Given such enumeration we define i) a sequence of databases (Γ_n, α_n), ii) a sequence of histories H_n, iii) a new enumeration of pairs (y_n, B_n), as follows:

- (step 0) Let $(\Gamma_0, \alpha_0) = (\Gamma_0, \alpha_0)$, $H_0 = H$, $(y_0, B_0) = (x, A)$.
- (step n+1) Given (y_n, B_n), if $y_n \in Lab(\Gamma_n)$ and $\Gamma_n, \alpha_n \ \vdash^? \ y_n : B_n, H_n$ fails then proceed according to (a) else to (b).
  - (a) if B_n if atomic, then we set

    $H_{n+1} = H_n \cup \{(y_n, B_n)\}$,
    $(\Gamma_{n+1}, \alpha_{n+1}) = (\Gamma_n, \alpha_n)$,
    $(y_{n+1}, B_{n+1}) = (x_{k+1}, A_{k+1})$,
    where $k = max_{t \leq n} \exists s_{\leq n} (y_s, B_s) = (x_t, A_t)$,

    else let $B_n = C \Rightarrow D$, then we set

    $H_{n+1} = H_n$,
    $(\Gamma_{n+1}, \alpha_{n+1}) = (\Gamma_n, \alpha_n) \otimes_{y_n} (x_m : C)$,
    $(y_{n+1}, B_{n+1}) = (x_m, D)$,
    where $x_m = min\{x_t \in \mathcal{A} \mid x_t \notin Lab(\Gamma_n)\}$.
  - (b) We set

    $H_{n+1} = H_n$,
    $(\Gamma_{n+1}, \alpha_{n+1}) = (\Gamma_n, \alpha_n)$,
    $(y_{n+1}, B_{n+1}) = (x_{k+1}, A_{k+1})$,
    where $k = max\{t \leq n \mid \exists s_{\leq n} (y_s, B_s) = (x_t, A_t)\}$,

Lemma 4.3.5 $\quad \forall k \ \exists n \geq k \ (x_k, A_k) = (y_n, B_n)$.

Proof. By induction on k. If $k = 0$, the claim hold by definition.
Let $(x_k, A_k) = (y_n = B_n)$.

- (i) if $y_n \notin Lab(\Gamma_n)$, or $\Gamma_n, \alpha_n \ \vdash^? \ y_n : B_n, H_n$ succeeds, or B_n is atomic, then $(x_{k+1}, A_{k+1}) = (y_{n+1}, B_{n+1})$.
- (ii) Otherwise, let $B_n = C_1 \Rightarrow \ldots \Rightarrow C_t \Rightarrow r$, $(t > 0)$, then

  $(x_{k+1}, A_{k+1}) = (y_{n+t+1}, B_{n+t+1})$.

□

Lemma 4.3.6 *For all $n \geq 0$, if $\Gamma_n, \alpha_n \vdash^? y_n : B_n, H_n$ fails, then:*

$\forall m \geq n$ $\Gamma_m, \alpha_m \vdash^? y_n : B_n, H_m$ *fails.*

Proof. By induction on $cp(B_n) = c$. if $c = 0$, that is B_n is an atom, say q, then we proceed by induction on $m \geq n + 1$.

- (m = n+1) we have $\Gamma_n, \alpha_n \vdash^? y_n : q, H_n$ fails, then also $\Gamma_n, \alpha_n \vdash^? y_n : q, H_n \cup \{(y_n, q)\}$ fails, whence, by construction,

  $\Gamma_{n+1}, \alpha_{n+1} \vdash^? y_n : q, H_{n+1}$ fails.

- (m ¿ n+1) Suppose we have proved the claim up to $m \geq n+1$, and suppose by way of contradiction that $\Gamma_m, \alpha_m \vdash^? y_n : q, H_m$ fails, but

  (i) $\Gamma_{m+1}, \alpha_{m+1} \vdash^? y_n : q, H_{m+1}$ succeeds.

  At step m, (y_m, B_m) is considered; it must be $y_m \in Lab(\Gamma_m)$ and

  (ii) $\Gamma_m, \alpha_m \vdash^? y_m : B_m, H_m$ fails.

  We have two cases, according to the form of B_m. If B_m is an atom r, as $(y_n, q) \in H_m$, from query (ii) by a restart we can step to

  $\Gamma_m, \alpha_m \vdash^? y_n : q, H_m \cup \{(y_m, r)\}$,

  that is the same as $\Gamma_{m+1}, \alpha_{m+1} \vdash^? y_n : q, H_{m+1}$, which succeeds and we get a contradiction.

If $B_m = C_1 \Rightarrow \ldots \Rightarrow C_t \Rightarrow r$, with $k > 0$, then from query (ii) we step in k steps to

$\Gamma_{m+k}, \alpha_{m+k} \vdash^? y_{m+k} : r, H_{m+k}$,

where $(\Gamma_{m+k}, \alpha_{m+k}) = (\Gamma_m, \alpha_m) \otimes_{y_m} (y_{m+1} : C_1) \otimes_{y_{m+1}} \ldots \otimes_{y_{m+k-1}} (y_{m+k} : C_k)$, and $H_{m+k} = H_m$, and then, by restart, since $(y_n, q) \in H_{m+k}$, to:

(iii) $\Gamma_{m+k}, \alpha_{m+k} \vdash^? y_n : q, H_{m+k} \cup \{(y_{m+k}, r)\}$,

Since query (i) succeeds, by monotony we have that also query (iii) succeeds, whence query (ii) succeeds, contradicting the hypothesis.

Let $cp(B_n) = c > 0$, that is $B_n = C \Rightarrow D$. By hypothesis

$\Gamma_n, \alpha_n \vdash^? y_n : C \Rightarrow D, H_n$, fails.

Then by construction, and computation rules

$\Gamma_{n+1}, \alpha_{n+1} \vdash^? y_{n+1} : D, H_{n+1}$, fails,

and hence, by induction hypothesis, $\forall m \geq n + 1$,

$\Gamma_m, \alpha_m \vdash^? y_{n+1} : D, H_m$, fails.

Suppose by way of contradiction that for some $m \geq n + 1$,

$\Gamma_m, \alpha_m \vdash^? y_n : C \Rightarrow D, H_m$, succeeds.

This implies that, for some $z \notin Lab(\Gamma_m)$,

(1) $(\Gamma_m, \alpha_m) \otimes_{y_n} (z : C) \vdash^? z : D, H_m$, succeeds.

Since $y_{n+1} : C \in \Gamma_{n+1} \subseteq \Gamma_m$, $\alpha_{n+1} \subseteq \alpha_m$, $H_{n+1} \subseteq H_m$, by monotony, we get

(2) $\Gamma_m, \alpha_m \vdash^? y_{n+1} : C, H_m$, succeeds.

The databases involved in queries (1) and (2) are clearly compatible for substitution, hence by cut we obtain that:

$\Gamma_m, \alpha_m \vdash^? y_{n+1} : D, H_m$ succeeds,

and we have a contradiction. □

Lemma 4.3.7 *(i)* $\forall m, \Gamma_m, \alpha_m \vdash^? x : A, H_m$ *fails;*
(ii) $\forall m$,*if* $(y, r) \in H_m$, *then* $\Gamma_m, \alpha_m \vdash^? y : r, H_m$ *fails.*

Proof. (i) is immediate by the previous lemma. To prove (ii), suppose it does not hold for some m and $(y, r) \in H_m$, i.e. $\Gamma_m, \alpha_m \vdash^? y : r, H_m$ succeeds. But then we can easily find a successful derivation of $\Gamma_m, \alpha_m \vdash^? x : A, H_m$ against (I); such derivation makes use of restart $(y, r) \in H_m$. □

Lemma 4.3.8 *If* $B_n = C \Rightarrow D$ *and*

$\Gamma_n, \alpha_n \vdash^? y_n : C \Rightarrow D, H_n$ *fails,*

then there is a $y \in \mathcal{A}$*, such that for* $k \leq n$*,* $y \notin Lab(\Gamma_k)$ *and* $\forall m > n$*:*

(i) $(y_n, y) \in \alpha_m$*,*
(ii) $\Gamma_m, \alpha_m \vdash^? y : C, H_m$ *succeeds,*
(iii) $\Gamma_m, \alpha_m \vdash^? y : D, H_m$ *fails.*

Proof. By construction, we can take $y = y_{n+1}$, the new point created at step $n + 1$, so that (i),(ii),(iii) hold for $m = n + 1$. In particular

(*) $\Gamma_{n+1}, \alpha_{n+1} \vdash^? y_{n+1} : D, H_{n+1}$ fails.

Since the (Γ, α_m) are not decreasing (wrt. inclusion), we immediately have that (i) and (ii) also hold for every $m > n + 1$. By construction, we know that $B_{n+1} = D$, whence by (*) and lemma 4.3.6, (iii) also holds for every $m > n + 1$. □

Construction of the Canonical model
We define an **S**-model as follows $M = (W, R, V)$, such that

- $W = \bigcup_n Lab(\Gamma_n)$;
- $xRy \equiv \exists n A^{\mathbf{S}}_{\alpha_n}(x, y)$,
-$V(x) = \{q \mid \exists n \; x \in Lab(\Gamma_n) \; \wedge \; \Gamma_n, \alpha_n \vdash^? x : q, H_n$ succeeds$\}$.

Lemma 4.3.9 *The relation* R *as defined above has the same properties of* $A^{\mathbf{S}}$*, e.g. if* **S**=**S4***, that is* $A^{\mathbf{S}}$ *is transitive and reflexive, then so is* R *and the same happens in all other cases.*

Proof. One easily verify the claim in each case. For instance, we verify that if $A^{\mathbf{S}}$ is euclidean (condition 5), then so is R. Assume xRy and xRz hold; by definition, there are n and m, such that $A^{\mathbf{S}}_{\alpha_n}(x,y)$ and $A^{\mathbf{S}}_{\alpha_m}(x,z)$. Let $k = max\{n,m\}$, since $A^{\mathbf{S}}$ is monotonic, we have both $A^{\mathbf{S}}_{\alpha_k}(x,y)$ and $A^{\mathbf{S}}_{\alpha_k}(x,z)$, and by property (5), also $A^{\mathbf{S}}_{\alpha_k}(z,y)$, whence zRy holds. □

Lemma 4.3.10 *for all $x \in W$ and formulas B,*

$$M, x \models B \;\Leftrightarrow\; \exists n\; x \in Lab(\Gamma_n) \wedge \Gamma_n, \alpha_n \;\vdash^?\; x : B, H_n \textit{ succeeds.}$$

Proof. We prove both directions by mutual induction on $cp(B)$. If B is an atom then the claim holds by definition. Thus, assume $B = C \Rightarrow D$.

($\Leftarrow$) Suppose for some m $\Gamma_m, \alpha_m \;\vdash^?\; x : C \Rightarrow D, H_m$ succeeds. Let xRy and $M, y \models C$, for some y. By definition of R, we have that for some n_1, $A_{\alpha_{n_1}}(x,y)$ holds. Moreover, by induction hypothesis, for some n_2, $\Gamma_{n_2}, \alpha_{n_2} \;\vdash^?\; y : C, H_{n_2}$ succeeds. Let $k = max\{n_1, n_2, m\}$, then we have

(1) $\Gamma_k, \alpha_k \;\vdash^?\; x : C \Rightarrow D, H_k$ succeeds,
(2) $\Gamma_k, \alpha_k \;\vdash^?\; y : C, H_k$ succeeds,
(3) $A^{\mathbf{S}}_{\alpha_k}(x,y)$.

So that from (1) we also have:

(1') $(\Gamma_k, \alpha_k) \otimes_x (z : C) \;\vdash^?\; z : D, H_k$ succeeds, (with $z \notin Lab(\Gamma_k)$).

We can cut (1') and (2), and obtain that:

$\Gamma_k, \alpha_k \cup \{(x,y)\} \;\vdash^?\; y : D, H_k$ succeeds.

Hence, by (3) and proposition 4.2.2(b) we get

$\Gamma_k, \alpha_k \;\vdash^?\; y : D, H_k$ succeeds,

and by induction hypothesis, $M, y \models D$,

($\Rightarrow$) Suppose by way of contradiction that $M, x \models C \Rightarrow D$, but for all n if $x \in Lab(\Gamma_n)$, then $\Gamma_n, \alpha_n \;\vdash^?\; x : C \Rightarrow D, H_n$ fails. Let $x \in Lab(\Gamma_n)$, then there are $m \geq k > n$, such that $(x, C \Rightarrow D) = (x_k, A_k) = (y_m, B_m)$ is considered at step $m+1$, so that we have:

$\Gamma_m, \alpha_m \;\vdash^?\; y_m : C \Rightarrow D, H_m$ fails.

By lemma 5.4.5, there is a $y \in \mathcal{A}$, such that (a) for $t \leq m$, $y \notin Lab(\Gamma_t)$ and (b):

$\forall m' > m$
(i) $(y_n, y) \in \alpha_{m'}$,
(ii) $\Gamma_{m'}, \alpha_{m'} \;\vdash^?\; y : C, H_{m'}$ succeeds,
(iii) $\Gamma_{m'}, \alpha_{m'} \;\vdash^?\; y : D, H_{m'}$ fails.

By (b)(i) we have xRy holds, by (b)(ii) and induction hypothesis, we have $M, y \models C$. By (a) and (b)(iii), we get that

$\forall n$ if $y \in Lab(\Gamma_n)$, then $\Gamma_n, \alpha_n \;\vdash^?\; y : D, H_n$ fails.

Hence, by induction hypothesis, we have $M, y \not\models D$, and we get a contradiction. □

Proof of The Completeness Theorem 4.3.4. We are able now to conclude the proof of the completeness theorem. Let $f(z) = z$, for every $z \in Lab(\Gamma)$, where $(\Gamma, \alpha) = (\Gamma_0, \alpha_0)$ is the original database. It is easy to see that f is a realization of (Γ, α) in M:

if $A^{\mathbf{S}}_{\alpha}(u, v)$ then $A^{\mathbf{S}}_{\alpha_0}(u, v)$, hence $f(u)Rf(v)$.

If $u : C \in \Gamma = \Gamma_0$, then by identity and the previous lemma we have $M, f(u) \models C$. On the other hand, by lemma 4.3.7, and the previous lemma we have $M, f(x) \not\models A$ and $M, f(y) \not\models r$ for every $(y, r) \in H$. This concludes the proof. ■

Corollary 4.3.11 *If $A \models_{\mathbf{S}} B$ holds, then $A \vdash^? x_0 : B, \emptyset$, succeeds in P($\mathbf{S}$) . In particular, if A is valid in the modal system S, then $\vdash^? x_0 : A, \emptyset$, succeeds in P($\mathbf{S}$) .*

4.4 Simplification for specific systems

In this section we show that for most modal logics we have considered, the use of labelled databases is not necessary and we can simplify either the structure of databases, either the deduction rules. We point out that the main results we have obtained for the general formulation with labels (cut-admissibility, completeness, restriction of the restart rule etc.) can be proved directly for each simplified system.

If we want to check the validity of a formula A, we evaluate A from a trivial database $\vdash^? x_0 : A, \emptyset$. Restricting our attention to computations from trivial databases, we observe that we can only generate database which have the form of trees.

Definition 4.4.1 A database (Δ, α) is called a *tree database* if the set of links α forms a tree.

Let (Δ, α) be a tree database and $x \in Lab(\Delta)$, we define the subdatabase $Path(\Delta, \alpha, x)$ as the *list* of labelled formulas lying on the path from the root of α, say x_0, up to x, that is: $Path(\Delta, \alpha, x) = (\Delta', \alpha')$, where:

$$\alpha' = \{(x_0, x_1), \dots, (x_{n-1}, x_n) \mid x_n = x \text{ and for } i = 1, \dots, n, (x_{i-1}, x_i) \in \alpha\}$$

$$\Delta' = \{y : A \in \Delta \mid y \in Lab(\alpha')\}.$$

■

Proposition 4.4.2 *If a query Q occur in any derivation from a trivial database, then $Q = \Delta, \alpha \vdash^? z : B, H$, where (Δ, α) is a tree database.*

From now on we restrict our consideration to tree-databases.

4.4.1 Simplification for K,K4,S4,T: databases as lists

We show that for systems **K**, **K4**, **S4**,**T**, the proof procedure can be simplified in the sense that: (i) the databases are lists of formulas, (ii) restart rule is not needed.

Given a successful query $Q = \Delta, \alpha \vdash^? z : B, H$ and a successful derivation $\mathcal{D}$ of Q, let

$\mathcal{R}(Q, \mathcal{D})$ = number of applications of the restart rule in $\mathcal{D}$.

Proposition 4.4.3 *If $Q = \Delta, \alpha \vdash^? x_1 : A_1, \{x_2 : A_2, \dots, x_k : A_k\}$ succeeds (with $A_2, \dots, A_k$ atoms) by a derivation $\mathcal{D}$ such that $\mathcal{R}(Q, \mathcal{D}) = m$, then, for* some $i = 1, \dots, k$, *the query*

$Q_i = Path(\Delta, \alpha, x_i) \vdash^? x_i : A_i, \emptyset$ *succeeds,*

by a derivation $\mathcal{D}_i$ *with* $\mathcal{R}(Q, \mathcal{D}_i) \leq m$.

Proof. Fix a derivation $\mathcal{D}$, with $\mathcal{R}(Q, \mathcal{D}) = m$; we proceed by induction on the height h of $\mathcal{D}$.

If $h = 0$, then A_1 is atomic and $x_1 : A \in \Delta$; since also $x_1 : A \in Path(\Delta, \alpha, x_1)$, we have that $Q_1 = Path(\Delta, \alpha, x_1) \vdash^? x_1 : A_1, \emptyset$ succeeds by a one-step derivation $\mathcal{D}_1$, with $\mathcal{R}(Q_1, \mathcal{D}_1) = 0$ and we are done.

Let $h > 0$. We have several cases according to what is the first deduction step of $\mathcal{D}$. Let $H = \{x_2 : A_2, \ldots, x_k : A_k\}$.

- (implication) In this case $A = B \Rightarrow C$ and the only child of Q is

  $(Q')\ \Delta', \alpha' \vdash^? y : C, H,$

  where $(\Delta', \alpha') = (\Delta, \alpha) \otimes_{x_1} (y : B)$, for $y \notin Lab(\Delta)$. We have that Q' succeeds by a subderivation D', with $\mathcal{R}(Q', \mathcal{D}') = m$, hence by induction hypothesis we have that either

  $(Q'')\ Path(\Delta', \alpha', y) \vdash^? y : C, \emptyset$ succeeds by a derivation $\mathcal{D}''$, $\mathcal{R}(Q'', \mathcal{D}'') \leq \mathcal{R}(Q', \mathcal{D}')$,
  or
  $(Q'_i)\ Path(\Delta', \alpha', x_i) \vdash^? x_i : A_i, \emptyset$ succeeds by a derivation $\mathcal{D}'_i$, $\mathcal{R}(Q'_i, \mathcal{D}'_i) \leq \mathcal{R}(Q', \mathcal{D}')$.

  In the former case, it is sufficient to append

  $(Q_1)\ Path(\Delta, \alpha, x_1) \vdash^? x_1 : B \Rightarrow C, \emptyset$

  on the top of $\mathcal{D}''$ to get the conclusion, since $Path(\Delta', \alpha', y) = Path(\Delta, \alpha, x_1) \otimes_{x_1} (y : C)$. In the latter case, we immediately conclude as $Path(\Delta', \alpha', x_i) = Path(\Delta, \alpha, x_i)$.

- (restart) In this case A_1 is atomic and the only child of Q is

  $(Q'_i)\ \Delta, \alpha \vdash^? x_i : A_i, H \cup \{(x_1, A_1)\},$

  which succeeds by a subderivation $\mathcal{D}'_i$ of $\mathcal{D}$. Hence, by induction hypothesis, for some $j = 1, \ldots, k$, $Q_j = Path(\Delta, \alpha, x_j) \vdash^? x_j : A_j, \emptyset$ succeeds by a derivation $\mathcal{D}_j$, with

  $\mathcal{R}(Q_j, \mathcal{D}_j) \leq \mathcal{R}(Q'_i, \mathcal{D}'_i) < \mathcal{R}(Q, \mathcal{D}).$

- (reduction) In this case, A_1 is an atom q and there is one $z : C \in \Delta$, with $C = B_1 \Rightarrow \ldots \Rightarrow B_t \Rightarrow q$, then Q has t children Q'_j, $j = 1, \ldots, t$:

  $(Q'_j)\ \Delta, \alpha \vdash^? z_j : B_j, H \cup \{(x_1, A_1)\},$

  such that $A^{\mathbf{S}}_\alpha(z, z_1), \ldots, A^{\mathbf{S}}_\alpha(z_{t-1}, z_t)$ hold, with $z_t = x_1$. Every Q'_j succeeds by a shorter derivation $\mathcal{D}'_j$, with

  $$\mathcal{R}(Q, \mathcal{D}) = \Sigma^t_j \mathcal{R}(Q_j, \mathcal{D}_j).$$

  By induction hypothesis, for each $j = 1, \ldots, t$, we have either

  (i) $Path(\Delta, \alpha, z_j) \vdash^? z_j : B_j, \emptyset$ succeeds, or
  (ii) $Path(\Delta, \alpha, x_i) \vdash^? x_i : A_i, \emptyset$ $(i > 1)$ succeeds, or
  (iii) $Path(\Delta, \alpha, x_1) \vdash^? x_1 : q, \emptyset$ succeeds

by a derivation $\mathcal{D}*_j$ whose $\mathcal{R}$ degree is no greater than the one of Q'_j. Clearly, if for some j either (ii) or (iii) holds we are done. Thus, suppose, that $j = 1, \ldots, t$, always (i) holds. Call, for each j, Q''_j the query involved in (i).

We show that $z, z_1, \ldots, z_{t-1}$ are on the path from the root x_0 to x_1. Let $z_0 = z, z_t = x_1$. It is sufficient to prove that for $j = 0, \ldots, t-1$, z_{t-j-1} is on the path from x_0, to z_{t-j}. Remember that it holds $A^{\mathbf{S}}_{\alpha}(z_{t-j-1}, z_{t-j})$. Since α is a tree, this means that either (a) z_{t-j-1} is the parent of z_{t-j}, or (b) $z_{t-j-1} = z_{t-j}$, or (c) z_{t-j-1} is an ancestor of z_{t-j}, (for **K** we have (a), for **T** (a) + (b), for **K4** (c), for **S4** (b) +(c)). In all cases, the claim follows as the path from the root to z_{t-j} is unique.

An immediate consequence of what we have shown is that for $j = 1, \ldots, t$, $Path(\Delta, \alpha, z_j) \subseteq Path(\Delta, \alpha, x_1)$, so that by monotony, for $j = 1, \ldots, t$,

(Q''_j) $Path(\Delta, \alpha, x_1) \vdash^? z_j : B_j, \emptyset$ succeeds,

and it is easy to see that it does so by a derivation $\mathcal{D}''_j$ with a $\mathcal{R}$-degree no greater than that of $\mathcal{D}*_j$. Since, $z : C \in Path(\Delta, \alpha, x_1)$, we obtain a derivation D_1 of $Q_1 = Path(\Delta, \alpha, x_1) \vdash^? x_1, q, \emptyset$ by making a tree with root Q_1 and appending to Q_1 each $\mathcal{D}''_j$. We observe that

$$\mathcal{R}(Q_1, \mathcal{D}_1) = \Sigma^t_j \mathcal{R}(Q''_j, \mathcal{D}''_j) \leq \Sigma^t_j \mathcal{R}(Q_j, \mathcal{D}_j) = \mathcal{R}(Q, \mathcal{D}).$$

□

Proposition 4.4.4 *Let $Q = \Delta, \alpha \vdash^? x : A, \emptyset$. If Q succeeds by a derivation $\mathcal{D}$ with $\mathcal{R}(Q, \mathcal{D}) = k$, then there is a successful derivation $\mathcal{D}'$ of Q with $\mathcal{R}(Q, \mathcal{D}) < k$.*

Proof. Suppose $k > 0$, inspecting $\mathcal{D}$ from the root downwards, we can find a query $Q_1 = \Delta, \beta \vdash^? y : q, H_1$, such that (y, q) is used in a restart step at some descendant, say Q', of Q_1. We say that Q' "makes use" of Q_1. Moreover, we can choose Q_1, such that no query in $\mathcal{D}$ makes use of an ancestor of Q_1 in a restart step. Let Q_2 be a descendant of Q_1 which makes use of it, and let Q_3 be the child of Q_2 obtained by restart, we have

$Q_2 = \Sigma, \gamma \vdash^? z : r, H_2,$
$Q_3 = \Sigma, \gamma \vdash^? y : q, H_2 \cup \{(z, r)\}.$

Let $\mathcal{D}_i$, $(i = 1, 2, 3)$, be the subderivation of $\mathcal{D}$ with root Q_i. Since no query in $\mathcal{D}_1$ makes use of an ancestor of Q_1, in a restart step, we obtain that the query $Q'_1 = \Delta, \beta \vdash^? y : q, \emptyset$, succeeds by a derivation $\mathcal{D}'_1$ obtained by removing history H_1 from any node of $\mathcal{D}_1$; Derivation $\mathcal{D}'_1$ will contain queries Q'_2 and Q'_3 corresponding to Q_2 and Q_3:

$Q'_2 = \Sigma, \gamma \vdash^? z : r, H_2 - H_1,$
$Q'_3 = \Sigma, \gamma \vdash^? y : q, (H_2 - H_1) \cup \{(z, r)\}.$

Let $\mathcal{D}'_3$ be the subderivation of $\mathcal{D}'_1$, with root Q'_3. We have that

(*) $\mathcal{R}(Q'_3, \mathcal{D}'_3) = \mathcal{R}(Q_3, \mathcal{D}_3) < \mathcal{R}(Q_2, \mathcal{D}_2) \leq \mathcal{R}(Q_1, \mathcal{D}_1)$.

By the previous proposition, one of the following (a), (b), (c) holds:

- (a) Query $Q_a = Path(\Sigma, \gamma, y) \vdash^? y : q, \emptyset$ succeeds by a derivation $\mathcal{D}_a$, such that $\mathcal{R}(Q_a, \mathcal{D}_a) \leq \mathcal{R}(Q'_3, \mathcal{D}'_3)$.

- (b) Query $Q_b = \ Path(\Sigma, \gamma, z) \ \vdash^? \ z : r, \emptyset$ succeeds by a derivation $\mathcal{D}_b$, such that $\mathcal{R}(Q_b, \mathcal{D}_b) \leq \mathcal{R}(Q'_3, \mathcal{D}'_3)$.
- (c) For some $(u_j, p_j) \in H_2 - H_1$, query $Q_c = \ Path(\Sigma, \gamma, u_j) \ \vdash^? \ u_j : p_j, \emptyset$ succeeds by a derivation $\mathcal{D}_c$, such that $\mathcal{R}(Q_c, \mathcal{D}_c) \leq \mathcal{R}(Q'_3, \mathcal{D}'_3)$. Moreover, there exists a query of the form

  $$Q'_j = \ \Delta_j, \gamma_j \ \vdash^? \ u_j, p_j, H_j \text{ in } \mathcal{D}'$$

  in the branch from Q'_1 to Q'_2 and there exists a corresponding query

  $$Q_j = \ \Delta_j, \gamma_j \ \vdash^? \ u_j, p_j, H_j \cup H_1$$

  occurring in $\mathcal{D}$ along the branch from Q_1 to Q_2.

In case (a) we have that $Path(\Sigma, \gamma, y) \subseteq (\Delta, \beta)$, hence by monotony, Q_1 succeeds by a derivation $\mathcal{D}'_a$, such that (by (*)),

$$\mathcal{R}(Q_1, \mathcal{D}'_a) = \mathcal{R}(Q_a, \mathcal{D}_a) < \mathcal{R}(Q_1, \mathcal{D}_1).$$

In case (b) we obviously have that $Path(\Sigma, \gamma, z) \subseteq (\Sigma, \gamma)$, hence by monotony, Q_2 succeeds by a derivation $\mathcal{D}'_b$, such that (by (*)),

$$\mathcal{R}(Q_2, \mathcal{D}'_b) = \mathcal{R}(Q_b, \mathcal{D}_b) < \mathcal{R}(Q_2, \mathcal{D}_2).$$

In case (c) we have that $Path(\Sigma, \gamma, u_j) \subseteq (\Delta_j, \gamma_j)$, hence by monotony, Q_j succeeds by a derivation $\mathcal{D}'_c$, such that $\mathcal{R}(Q_j, \mathcal{D}'_c) = \mathcal{R}(Q_c, \mathcal{D}_c)$. If we call $\mathcal{D}_j$ the subderivation of $\mathcal{D}$ with root Q_j, we have $\mathcal{R}(Q_j, \mathcal{D}_j) > \mathcal{R}(Q_3, \mathcal{D}_3)$, and hence (by (*)),

$$\mathcal{R}(Q_j, \mathcal{D}'_c) = \mathcal{R}(Q_c, \mathcal{D}_c) \leq \mathcal{R}(Q'_3, \mathcal{D}'_3) = \mathcal{R}(Q_3, \mathcal{D}_3) < \mathcal{R}(Q_j, \mathcal{D}_j).$$

According to the case (a), (b), (c), we obtain a successful derivation $\mathcal{D}'$, with $\mathcal{R}(Q, \mathcal{D}') < \mathcal{R}(Q, \mathcal{D})$, by replacing in $\mathcal{D}$, either $\mathcal{D}_1$ by $\mathcal{D}'_a$, or $\mathcal{D}_2$ by $\mathcal{D}'_b$, or $\mathcal{D}_j$ by $\mathcal{D}'_c$. □

By repeatedly applying the previous proposition, we get

Theorem 4.4.5 *If $\Delta, \alpha \ \vdash^? \ x : A, \emptyset$ succeeds, then $Path(\Delta, \alpha, x) \ \vdash^? \ x : A, \emptyset$ succeeds without using restart.*

By virtue of this theorem we can reformulate the proof system for logics from **K** to **S4** as follows. A database is simply a list of formulas $A_1, \ldots, A_n$, which stands for the labelled database $(\{x_1 : A_1, \ldots, x_n : A_n\}, \alpha)$, where $\alpha = \{(x_1, x_2), \ldots (x_{n-1}, x_n)\}$. A query has the form:

$$A_1, \ldots, A_n \ \vdash^? \ B$$

which stands for $\{x_1 : A_1, \ldots, x_n : A_n\}, \alpha \ \vdash^? \ x_n : B$. The history has been omitted since restart is not needed. Letting $\Delta = A_1, \ldots, A_n$, we reformulate the predicates $A^{\mathbf{S}}$ as relations between formulas within a database $A^{\mathbf{S}}(\Delta, A_i, A_j)$, in particular we can define:

$$A^K(\Delta, A_i, A_j) \equiv i + 1 = j$$
$$A^T(\Delta, A_i, A_j) \equiv i = j \vee i + 1 = j$$
$$A^{\mathbf{K4}}(\Delta, A_i, A_j) \equiv i < j$$
$$A^{\mathbf{S4}}(\Delta, A_i, A_j) \equiv i \leq j$$

The rules become as follows:

- (success) $\Delta \vdash^? q$ succeeds if $\Delta = A_1, \ldots, A_n$, and $A_n = q$;
- (implication) from

  $\Delta \vdash^? A \Rightarrow B$

  step to

  $\Delta, A \vdash^? B$;
- (reduction) from

  $\Delta \vdash^? q$

  step to

  $\Delta_i \vdash^? D_i$, for $i = 1, \ldots, k$,

  if there is a formula $A_j = D_1 \Rightarrow \ldots \Rightarrow D_k \Rightarrow q \in \Delta$, and there are integers $j = j_0 \leq j_1 \leq \ldots \leq j_k = n$, such that

  $i = 1, \ldots, k$, $A^{\mathbf{S}}(\Delta, A_{j_{i-1}}, A_{j_i})$ holds and $\Delta_i = A_1, \ldots, A_{j_i}$.

Example 4.4.6 We show that $((b \Rightarrow a) \Rightarrow b) \Rightarrow c \Rightarrow (b \Rightarrow a) \Rightarrow a$ is a theorem of **S4**.

$$\begin{array}{rcll}
 & \vdash^? & ((b \Rightarrow a) \Rightarrow b) \Rightarrow c \Rightarrow (b \Rightarrow a) \Rightarrow a & \\
(b \Rightarrow a) \Rightarrow b & \vdash^? & c \Rightarrow (b \Rightarrow a) \Rightarrow a & \\
(b \Rightarrow a) \Rightarrow b, c & \vdash^? & (b \Rightarrow a) \Rightarrow a & \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a & \vdash^? & a & \text{reduction wrt. } b \Rightarrow a \ (1) \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a & \vdash^? & b & \text{reduction wrt. } (b \Rightarrow a) \Rightarrow b \ (2) \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a & \vdash^? & b \Rightarrow a & \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a, b & \vdash^? & a & \text{reduction wrt. } b \Rightarrow a \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a, b & \vdash^? & b &
\end{array}$$

This formula fails in both **T** and **K4**, whence in **K**: reduction on step (1) is allowed in **T** but not in **K4**; on the contrary, reduction on step (2) is allowed in **K4** but not in **T**.

■

Disjunction Property
To conclude this section, we observe that the previous results and in particular proposition 4.4.3 can be used to prove a *disjunction property* for modal logics **S** ranging from **K** to **S4**. The property is the following: let $A, B \in \mathcal{L}(\Rightarrow)$, then we have:

if $\models_{\mathbf{S}} A \vee B$, then either $\models_{\mathbf{S}} A$ or $\models_{\mathbf{S}} b$.

This property holds for modal logics from **K** to **S4**, but it does not for the other systems we have considered so far (for instance in **S5**, we have $\models (p \Rightarrow q) \vee ((p \Rightarrow q) \Rightarrow r)$, but $\not\models p \Rightarrow q$ and $\not\models (p \Rightarrow q) \Rightarrow r$). To show this property, we first observe that both the completeness theorem and proposition 4.4.3 can be extended without any effort to more general queries in which the history may contain non-atomic formulas of $\mathcal{L}(\Rightarrow)$, rather than mere atoms. The only change in the rules is that in a restart step, we are allowed to select a pair (x, C) from the history H even if C is non-atomic. The semantic meaning of a query remains the same as before. With this extension, we can show the disjunction property as follows. If $\models_{\mathbf{S}} A \vee B$, then by completeness we have that the query

$$\vdash^{?} \ x_0 : A, \{(x_0, B)\} \text{ succeeds in } \mathrm{P}(\mathbf{S}).$$

By proposition 4.4.3 either $\vdash^{?} \ x_0 : A, \emptyset$, or $\vdash^{?} \ x_0 : B, \emptyset$ succeeds. By the soundness of the proof-procedure, we get that either $\models_{\mathbf{S}} A$ or $\models_{\mathbf{S}} B$. ■

4.4.2 Simplification for K5, K45, S5: databases as clusters

In this section we give an unlabelled formulation of logics **K5**, **K45**, **S5**. In the following proposition by **S** we intend **K5**, or **K45**.

Proposition 4.4.7 *Let $Q = \Gamma, \alpha \ \vdash^{?} \ x : G, H$ be any query which occurs in a deduction from a trivial database $x_0 : A \ \vdash^{?} \ x_0 : B, H_0$, then*

$$\forall z \in \mathit{Lab}(\alpha), \neg A^{\mathbf{S}}_{\alpha}(z, x_0).$$

Proof. We give the proof for **K5** only, for **K45** is very similar. By proposition 4.4.2, α is a tree with root x_0, thus if $(x_0, v) \in \alpha$, then for all descendant z of x_0 (that is for all $z \in \mathit{Lab}(\alpha)$), we have

$$(*) \ (z, x_0) \notin \alpha.$$

By definition, we know that $A^{\mathbf{K5}}$ is the least euclidean closure of α. By a standard argument, we have:

$$A^{\mathbf{S}}_{\alpha}(x, y) \Leftrightarrow \exists n A^{\mathbf{S}}_{n,\alpha}(x, y), \text{ where}$$
$$A^{\mathbf{S}}_{0,\alpha}(x, y) \equiv (x, y) \in \alpha,$$
$$A^{\mathbf{S}}_{n+1,\alpha}(x, y) \equiv A^{\mathbf{S}}_{n,\alpha}(x, y) \vee \exists z (A^{\mathbf{S}}_{n,\alpha}(z, x) \wedge A^{\mathbf{S}}_{n,\alpha}(z, y)).$$

Now, we show by induction on n, that $\forall n \ \neg A^{\mathbf{S}}_{n,\alpha}(z, x_0)$. For $n = 0$, it holds by (*). Assume the claim holds for n; if $A^{\mathbf{S}}_{n+1,\alpha}(y, x_0)$ held for some y, then by induction hypothesis, we would get that there exists one z such that $A^{\mathbf{S}}_{n,\alpha}(z, x_0) \wedge A^{\mathbf{S}}_{n,\alpha}(z, y))$, contrary to the induction hypothesis. □

Proposition 4.4.8 *Let $Q = \Delta, \beta \ \vdash^{?} \ x : G, H$ be any query which occurs in a deduction from a trivial database $x_0 : A \ \vdash^{?} \ x_0 : B, H_0$,*

if $(x_0, y_1), (x_0, y_2) \in \beta$, then $y_1 = y_2$.

Proof. The proof is the same for both **K5** and **K45**. Fix a deduction $\mathcal{D}$. Assume by way of contradiction that $(x_0, y_1), (x_0, y_2) \in \beta$, but $y_1 \neq y_2$, for some query $Q = \Delta, \beta \ \vdash^{?} \ x : G, H$ appearing in $\mathcal{D}$. Since at the root of $\mathcal{D}$ the graph is empty, points y_1, y_2, or better, links $(x_0, y_1), (x_0, y_2)$ have been created because of the evaluation of some queries which are ancestors of Q. Without loss of generality we can assume that y_1 has been introduced before y_2. That is, above Q, in the same branch, there is a query

$$Q' : \ \Gamma, \beta'_1 \ \vdash^{?} \ x_0 : C \Rightarrow D, H',$$

whose child is

$$Q'' : \ (\Gamma, \beta') \otimes_{x_0} (y_2 : C) \ \vdash^? \ y_2 : D, H',$$

with $(x_0, y_1) \in \beta'$ and $\beta' \cup \{(x_0, y_2)\} \subseteq \beta$. Query Q is Q'' or is one of its descendants. Since $\beta' \neq \emptyset$, Q' cannot be the root of $\mathcal{D}$; it is easy to see that Q' cannot be obtained neither by the implication rule (x_0 is not a new point), nor by restart ($C \Rightarrow D$ is not atomic). Hence the only possibility is that Q' has been obtained by reduction wrt some $z : C \in \Gamma$, with $C = F_1 \Rightarrow \ldots \Rightarrow F_k \Rightarrow q$, from a query $Q_0 = \ \Gamma, \beta_1' \ \vdash^? \ u : q, H''$. Thus, there are some u_i, for $i = 1, \ldots, k-1$, such that letting $u_0 = z$, and $u_k = u$, $A^{\mathbf{S}}_{\beta}(u_{i-1}, u_i)$ holds, and it must be $u_j = x_0$ and $F_j = C \Rightarrow D$, for some $1 \leq j \leq k$. But this implies that $A^{\mathbf{S}}_{\beta'}(u_{j-1}, x_0)$ holds, contradicting the previous proposition. □

Proposition 4.4.9 *Let $Q = \Delta, \alpha \ \vdash^? \ x : G, H$ be any query which occurs in a P(**K5**) deduction from a trivial database $x_0 : A \ \vdash^? \ x_0 : B, H_0$. Let $R^{\mathbf{K5}}_{\alpha}(x,y)$ be defined as follows:*

$$R^{\mathbf{K5}}_{\alpha}(x,y) \equiv (x = x_0 \wedge (x_0, y) \in \alpha) \ \vee \ (x, y \in Lab(\alpha) \wedge x \neq x_0 \wedge y \neq x_0)$$

Then we have $R^{\mathbf{K5}}_{\alpha}(x,y) \equiv A^{\mathbf{K5}}_{\alpha}(x,y)$.

Proof. ($\Rightarrow$) By induction on $\mid \alpha \mid$, we show that $R^{\mathbf{K5}}_{\alpha}(x,y) \rightarrow A^{\mathbf{K5}}_{\alpha}(x,y)$. If $\alpha = \emptyset$, then $R^{\mathbf{K5}}_{\alpha}(x,y)$ does not hold and the claim trivially follows. Let $\alpha = \{(x_0, v)\}$ with $v \neq x_0$, then

$$R^{\mathbf{K5}}_{\alpha}(x,y) \equiv (x = x_0 \wedge y = v) \ \vee \ (x = y \wedge y = v)$$

In the first case, $(x = x_0 \wedge y = v)$, we conclude that $A^{\mathbf{K5}}_{\alpha}(x,y)$, since $(x,y) \in \alpha \rightarrow A^{\mathbf{K5}}_{\alpha}(x,y)$; in the latter case, by $A^{\mathbf{K5}}_{\alpha}(x_0, v)$ and the euclidean property we get $A^{\mathbf{K5}}_{\alpha}(v,v)$.

Let $\mid \alpha \mid > 1$, so that $\alpha = \alpha' \cup \{(u,v)\}$, with $u \neq x_0$, notice that it must be also $v \neq x_0$. It is easy to see that

$$R^{\mathbf{K5}}_{\alpha}(x,y) \equiv R^{\mathbf{K5}}_{\alpha'}(x,y) \ \vee \ (x,y) = (u,v).$$

Thus, if $R^{\mathbf{K5}}_{\alpha'}(x,y)$ holds, then by induction hypothesis, we get $A^{\mathbf{K5}}_{\alpha'}(x,y)$, and by monotony also $A^{\mathbf{K5}}_{\alpha}(x,y)$; if $(x,y) = (u,v)$, we conclude by the fact that $(x,y) \in \alpha \rightarrow A^{\mathbf{K5}}_{\alpha}(x,y)$.

($\Leftarrow$) First we check that if $(x,y) \in \alpha$, then $R^{\mathbf{K5}}_{\alpha}(x,y)$. To this regard let $(x,y) \in \alpha$, if $x = x_0$, and $(x_0, y) \in \alpha$, then $R^{\mathbf{K5}}_{\alpha}(x,y)$, otherwise if $x \neq x_0$, then by proposition 4.4.7, $y \neq x_0$, and hence by definition $R^{\mathbf{K5}}_{\alpha}(x,y)$. Next we check that $R^{\mathbf{K5}}_{\alpha}(x,y)$ and $R^{\mathbf{K5}}_{\alpha}(x,z)$ imply $R^{\mathbf{K5}}_{\alpha}(x,z)$. To this purpose, assume $x = x_0$, then by the previous propositions, we get $y = z \neq x_0$, and hence by definition $R^{\mathbf{K5}}_{\alpha}(x,z)$; if $x \neq x_0$, then by proposition 4.4.7, we get $y \neq x_0$, and $z \neq x_0$, and hence by definition $R^{\mathbf{K5}}_{\alpha}(x,z)$. Then we conclude by the minimality of $A^{\mathbf{K5}}$. □

Corollary 4.4.10 *At the same conditions as the last proposition, we have:*

$$R^{\mathbf{K5}}_{\alpha}(x_0, x) \wedge R^{\mathbf{K5}}_{\alpha}(x_0, y) \Rightarrow x = y.$$

Proposition 4.4.11 *Let $Q = \Delta, \alpha \ \vdash^? \ x : G, H$ be any query which occurs in a P(**K45**) deduction from a trivial database $x_0 : A \ \vdash^? \ x_0 : B, H_0$. Let $R^{\mathbf{K45}}_{\alpha}(x,y)$ be defined as follows:*

$$R^{\mathbf{K45}}_{\alpha}(x,y) \equiv x, y \in Lab(\alpha) \wedge y \neq x_0.$$

Then we have $R^{\mathbf{K45}}_{\alpha}(x,y) \equiv A^{\mathbf{K45}}_{\alpha}(x,y)$.

Proof. Similar to that one of the previous proposition, and hence left to the reader. □

Proposition 4.4.12 *Let $Q = \Delta, \alpha \vdash^? x : G, H$ be any query which occurs in a P(***S5***) deduction from a trivial database $x_0 : A \vdash^? x_0 : B, H_0$. Let $R^{\mathbf{S5}}_{\alpha}(x,y)$ be defined as follows:*

$$R^{\mathbf{S5}}_{\alpha}(x,y) \equiv x, y \in Lab(\alpha)$$

Then we have $R^{\mathbf{S5}}_{\alpha}(x,y) \equiv A^{\mathbf{S5}}_{\alpha}(x,y)$.

Proof. We observe that $A^{\mathbf{S5}}$ is the equivalence relation which contains just one class, that one of the labels which are descendant of x_0, that is all labels occurring in α. □

By virtue of the previous propositions we can reformulate the proof systems for **K5**, **K45** and **S5** without making use of labels. For **K5** the picture is as follows, either a database contains just one point x_0, or there is an initial point x_0 which is connected only to another point x_1, and any point excluding x_0 is connected with any other including itself. In case of **K45**, x_0 is connected also to any other point different from x_0. Thus, in order to get a concrete structure without labels we must distinguish, the initial world, represented by x_0, the only point to which it is connected, say x_1 (in case of **K5**), and the current point in which the goal formula is evaluated.

A non-empty database has the form:

$$\Delta = B_0 \mid \quad \mid \text{ or } \Delta = B_0 \mid B_1, \ldots, B_n \mid B_i, \text{ where } 1 \leq i \leq n,$$

and $B_0, B_1, \ldots, B_n$ are formulas. We also define

$$Actual(\Delta) = \begin{cases} B_0 \text{ if } \Delta = B_0 \mid \quad \mid, \\ B_i \text{ if } \Delta = B_0 \mid B_1, \ldots, B_n \mid B_i \end{cases}$$

This rather odd structure is forced by the fact that in **K5** and **K45** we have reflexivity in all worlds, but in the initial one, whence differently from all other systems we have considered so far, the success of

$\vdash^? x_0 : A \Rightarrow B$, which means that $A \Rightarrow B$ is valid,

does not imply the success of

$x_0 : A \vdash^? x_0 : B$, which means that $A \rightarrow B$ is valid (material implication) [4].

The addition operation is defined as follows:

$$\Delta \otimes A = \begin{cases} B_0 \mid B_1, \ldots, B_n, B \mid A \quad \text{if } \Delta = B_0 \mid B_1, \ldots, B_n \mid B_i \\ true \mid A \mid A \quad \text{if } \Delta = \emptyset \end{cases}$$

A query has the form

$\Delta \vdash^? G, H$, where $H = \{(A_1, q_1), \ldots, (A_k, q_k)\}$, with $A_j \in \Delta$.

Definition 4.4.13 [Deduction Rules for **K5** and **K45**] Given $\Delta = B_0 \mid B_1, \ldots, B_n \mid B$, let

$A^5(\Delta, X, Y) \equiv (X = B_0 \wedge Y = B_1) \vee (X = B_i \wedge Y = B_j \wedge i, j > 0)$ and
$A^{45}(\Delta, X, Y) \equiv (X \in \Delta \wedge Y = B_j \text{ with } j > 0)$

- (success) $\Delta \vdash^? q, H$ succeeds if $Actual(\Delta) = q$.

[4] In this two systems the validity of $\Box C$ does not imply the validity of C, as it holds for all the other systems considered in this chapter.

- (implication) From $\Delta \vdash^? A \Rightarrow B, H$ step to $\Delta \otimes A \vdash^? B, H$.
- (reduction) if $\Delta = B_0 \mid B_1, \ldots, B_n \mid B$ and $C = D_1 \Rightarrow \ldots \Rightarrow D_k \Rightarrow q \in \Delta$, then from $\Delta \vdash^? G, H$ step to

  $B_0 \mid B_1, \ldots, B_n \mid C_i \vdash^? D_i, H \cup \{(B, q)\}$, for $i = 1, \ldots, k$,

  provided, letting $C_0 = C$, and $C_k = B_n$, $A^5(\Delta, C_{i-1}, C_i)$ (respectively $A^{45}(\Delta, C_{i-1}, C_i)$) holds.
- (restart) If $\Delta = B_0 \mid B_1, \ldots, B_n \mid B_i$ and $(B_j, r) \in H$, with $j > 0$, then from $\Delta \vdash^? q, H$, step to

  $B_0 \mid B_1, \ldots, B_n \mid B_j \vdash^? r, H \cup \{(B_i, q)\}$,

■

According to the above discussion, we observe that the check of the validity of $\models A \Rightarrow B$, corresponds to the query

$$\emptyset \vdash^? A \Rightarrow B, \emptyset,$$

which (by the implication rule) is reduced to the query

$$\mathit{true} \mid A \mid A \vdash^? B, \emptyset.$$

This is different from checking the validity of $A \rightarrow B$ ($\rightarrow$ is the material implication), which corresponds to the query

$$A \mid \quad \mid \vdash^? B, \emptyset.$$

The success of the latter query does not imply the success of the former. For instance the in **K5**,

$\not\models (\mathit{true} \Rightarrow p) \rightarrow p$ and indeed $\mathit{true} \Rightarrow p \mid \quad \mid \vdash^? p, \emptyset$ fails.

On the other hand we have

$\models (\mathit{true} \Rightarrow p) \Rightarrow p$ and indeed $\mathit{true} \mid \mathit{true} \Rightarrow p \mid \mathit{true} \Rightarrow p \vdash^? p, \emptyset$ succeeds.

The reformulation of the proof system for **S5** is similar, but simpler. In case of **S5**, there is no need to keep separate the first formula/world from the others. Thus, we may simply define a non-empty database as a pair $\Delta = (S, A)$, where S is a set of formulas and $A \in S$. If $\Delta = (S, A)$, we let

$$\mathit{Actual}(\Delta) = A, \text{ and } \Delta \otimes B = (S \cup \{B\}, B).$$

For $\Delta = \emptyset$, we define $\emptyset \otimes A = (\{A\}, A)$. With this definitions the rules are similar to those for **K5** and **K45**, with the following simplifications:

- (reduction) if $\Delta = (S, B)$ and $C = D_1 \Rightarrow \ldots \Rightarrow D_k \Rightarrow q \in \Delta$, then from $\Delta \vdash^? G, H$ step to

  $(S, C_i) \vdash^? D_i, H \cup \{(B, q)\}$, where $C_i \in \Delta$, for $i = 1, \ldots, k$,

  provided $C_k = B$.
- (restart) If $\Delta = (S, B)$ and $(C, r) \in H$, then from $(S, B) \vdash^? q, H$, step to

  $(S, C) \vdash^? r, H \cup \{(B, q)\}$,

Example 4.4.14 In Figure 4.4 we show a derivation of the following formula in **S5**

$$((a \Rightarrow b) \Rightarrow c) \Rightarrow (a \Rightarrow d \Rightarrow c) \Rightarrow (d \Rightarrow c)$$

In the derivation we make use of restricted restart, according to proposition 4.1.6. A brief explanation of the derivation: step (5) is obtained by reduction wrt $(a \Rightarrow b) \Rightarrow c$, step (7) by restart, steps (8) and (9) by reduction with respect to $a \Rightarrow d \Rightarrow c$, and they both succeed immediately.

$$\begin{tikzcd}
 & (1)\ \emptyset \vdash^? ((a \Rightarrow b) \Rightarrow c) \Rightarrow (a \Rightarrow d \Rightarrow c) \Rightarrow d \Rightarrow c \arrow[d, dash] & \\
 & (2)\ \{(a \Rightarrow b) \Rightarrow c\}, (a \Rightarrow b) \Rightarrow c \vdash^? (a \Rightarrow d \Rightarrow c) \Rightarrow (d \Rightarrow c) \arrow[d, dash] & \\
 & (3)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c\}, a \Rightarrow d \Rightarrow c \vdash^? d \Rightarrow c \arrow[d, dash] & \\
 & (4)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d\}, d \vdash^? c \arrow[d, dash] & \\
 & (5)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d\}, d \vdash^? a \Rightarrow b, (\mathbf{d}, \mathbf{c}) \arrow[d, dash] & \\
 & (6)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d, a\}, a \vdash^? b, (\mathbf{d}, \mathbf{c}) \arrow[d, dash] & \\
 & (7)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d, a\}, d \vdash^? c, (\mathbf{d}, \mathbf{c}) \arrow[dl, dash] \arrow[dr, dash] & \\
(8)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d, a\}, a \vdash^? a, (\mathbf{d}, \mathbf{c}) & & (9)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d, a\}, d \vdash^? d, (\mathbf{d}, \mathbf{c})
\end{tikzcd}$$

Figure 4.4:

■

4.5 An intuitionistic version of K5, K45, S5, B, BT

We have seen that if we remove the restart rule from the proof systems for **K**,**T**,**K4**, and **S4**, the relative proof system retains its completeness. The same does not hold for the other systems: if we remove the restart rule we obtain weaker proof systems and fewer theorems. On the other hand, the proof procedures even without restart are well-defined. From a proof-theoretical point of view they make sense and they enjoy several properties, the most important of them is the admissibility of cut as formulated in section 4.2. It is natural to wonder if there is a 'logic' which corresponds to these proof-systems. Here by a logic, we intend a semantical characterization of the set of successful formulas under each proof system. What we show in this section is that what we get, by removing the restart rule is essentially an intuitionistic version of the respective classical modal logics **K5**, **K45**, **S5**, **KB**, **KBT**.

We begin with a semantical presentation of the intuitionistic modal logics we take into account. The construction we present falls under the general methodology for combining logic presented in

[Gabbay 96]. A similar construction, for tense logics has been presented by Ewald in [Ewald 86]. The semantic structure introduced in the next definition can be seen as a generalization of the possible-world semantics of intuitionistic and modal logic.

Definition 4.5.1 Let **S** be any modal system, a model M for its intuitionistic version **SI**, has the form

$$M = (W, \leq, S^w, R^w, h^w)$$

where W is a non-empty set whose element are called *states*, $\leq$ is a transitive-reflexive relation on W, for each $w \in W$, the triple (S^w, R^w, h^w) is a Kripke model, that is S^w is a non-empty set, R^w is a binary relation on S^w having the properties of the accessibility relation of the corresponding system S, h^w map each $x \in S^w$ into a set of propositional variables. We assume the following monotony conditions:

1. if $w \leq w'$ then $S^w \subseteq S^{w'}$;
2. $w \leq w'$ then $R^w \subseteq R^{w'}$;
3. if $w \leq w'$ and $x \in S^w$ then $h^w(x) \subseteq h^{w'}(x)$.

Truth conditions for $\mathcal{L}(\Rightarrow)$ are defined as follows:

$S^w, x \models p$ if $p \in h^w(x)$;
$S^w, x \models A \Rightarrow B$ if for all $w' \geq w$ and for all $y \in S^{w'}$, such that $R^{w'}(x, y)$ it holds:

$$S^{w'}, y \models A \text{ implies } S^{w'}, y \models B.$$

We can easily see that for every formula A and $x \in S^w$:

if $S^w, x \models A$ and $w \leq w'$ then $S^{w'}, x \models A$,

We say that A is valid in an **SI**-model $M = (W, \leq, S^w, R^w, h^w)$ if $\forall w \in W$, $\forall y \in S^w$ $S^w, y \models A$. As usual, a formula is called **SI**-*valid* if it is valid in every **SI**-model. ■

As we have said, that the above semantical specification generalizes the possible-world semantics of both intuitionistic and modal logic. The structures introduced in the previous definition can be seen as models of intuitionistic logic in which single worlds are replaced by Kripke models. Thus, if for any w, S^w contains only one world, say $\{x_w\}$ and $R^w = \{(x_w, x_w)\}$, then we have a standard Kripke model of intuitionistic logic (see definition 2.1.1); on the other hand if W contains only one state, we have a standard Kripke model of modal logics as defined at the beginning of the chapter.

It can be shown that the proof procedures without restart are sound and complete for each **SI** $\in \{$**K5I**, **K45I**, **S5I**, **KBI**, **KBTI**$\}$.

Theorem 4.5.2 *A is valid in* **SI** *iff* $\vdash^? x_0 : A, \emptyset$ *succeeds in the proof system for $P(\mathbf{S})$, without using restart.*

Proof. The soundness is proved as usual by induction on the lenght of successful derivations. We leave the details to the reader.

With regard to the completeness, we can define a canonical model $M^{\mathbf{SI}}$ as follows:

$M^{\mathbf{SI}} = (W, S^w, h^w, R^w, \leq)$, where W is the set of finite non-empty databases (Γ, α), for $w = (\Gamma, \alpha) \in W$, $\leq$ is $\subseteq$, S^w is $Lab(\Gamma)$, $R^w(x, y) \equiv A^{\mathbf{S}}_{\alpha}(x, y)$ and h^w is defined as follows: for $x \in w$,

$$h^w(x) = \{q : \ w \ \vdash^? \ x : q \text{ succeeds}\}.$$

Notice that M satisfies all the monotony conditions involved in the semantics. We can easily prove that $M^{\mathbf{SI}}$ satisfies the property:

for all $w \in W, x \in S^w$, and formulas C, we have

$$w, x \models C \ \Leftrightarrow \ w \ \vdash^? \ x : C \text{ succeeds.}$$

By this fact completeness follows immediately. □

We conclude this section with an observation on the relationship between the classical modal logics and their respective intuitionistic version. Let us identify any system with the set of its valid formulas. We may observe that although the semantic of the intuitionistic modal logic is a combination of the modal and the intuitionistic possible-world semantics, each intuitionistic modal logic is strictly weaker than the intersection of intuitionstic logic **I** and the corresponding classical modal logic.

Proposition 4.5.3 *Let* **S** *be any one of* **K5**, **K45**, **S5**, **KB**, **KBT**, *and let* **SI** *be its intuitionistic version as defined above, we have*

$$\mathbf{SI} \subset \mathbf{I} \cap \mathbf{S}.$$

Proof. The claim $\mathbf{SI} \subseteq \mathbf{I} \cap \mathbf{S}$ easily follows from the previous theorem. To show that the inclusion is proper, consider the formula:

$$(((p \Rightarrow q) \Rightarrow q) \Rightarrow q) \Rightarrow p \Rightarrow q.$$

One can check that this formula is a theorem of **I** and of any **S**; however it is not a theorem of **S5I**, whence it is not theorem of any of the other (weaker) **SI**. □

4.6 Extending the language

In this section we extend the proof procedures presented to a broader fragment. We first consider a simple extension allowing conjunction, then we consider a richer fragments containing 'local clauses' and $(\wedge, \vee)$-combinations of goals. We do not have the pretence of finding a maximal fragment of modal logics which allows a goal-directed treatment and we just suggest a straightforward extension of the proof-methods we have described, which does not need any additional machinery.

4.6.1 Conjunction

To handle conjunction in the labelled formulation, we simply drop the condition that a label x may be attached to only one formula, hence formulas with the same label can be thought as logically conjuncted. In the unlabelled formulation, for those systems enjoying such a formulation, the general principle is to deal with *sets* of formulas, instead of *single formulas*. Databases will be structured collection of *sets* of formulas, rather than collection of formulas. The structure is always the same, but the constituents are now sets. Thus, in case of **K**, **T**, **K4** and **S4**, databases will be lists of *sets* of formulas, whereas in case of **K5**, **K45** and **S5**, will be clusters of *sets* of formulas.

A conjunction of formulas is interpreted as a set, so that queries may contain *sets* of goal formulas.

A formula A of language $\mathcal{L}(\wedge, \Rightarrow)$ is in *normal form* if it is an atom or has form:

$$\bigwedge_i [S^i_1 \Rightarrow \ldots \Rightarrow S^i_{n_i} \Rightarrow q_i]$$

where S^i_j are conjunctions of formulas in *normal form*. In all modal logics considered in this chapter (formulated $\mathcal{L}(\wedge, \Rightarrow)$) it holds that every formula has an equivalent one in normal form.

Proposition 4.6.1 *Let A be a formula in $\mathcal{L}(\wedge, \Rightarrow)$ then there is a formula $NF(A)$ in normal form such that $\models_{\mathbf{K}} A \leftrightarrow NF(A)$.*

Proof. By a straightforward induction on the structure of A. $\square$

We simplify the notation for NF formulas and replace conjunctions with sets. For example the NF of

$$(b \Rightarrow (c \wedge d)) \Rightarrow (e \wedge f)) \wedge ((g \wedge h) \Rightarrow (k \wedge u))$$

is the set containing the following formulas:

$$\big\{\{b \Rightarrow c, b \Rightarrow d\} \Rightarrow e, \{b \Rightarrow c, b \Rightarrow d\} \Rightarrow f, \{g, h\} \Rightarrow k, \{g, h\} \Rightarrow u\big\}.$$

For the deduction procedures all we have to do is to handle sets of formulas. We define, for $x \in Lab(\Gamma)$, $y \notin Lab(\Gamma)$ and finite set of formulas $S = \{D_1, \ldots, D_t\}$,

$(\Gamma, \alpha) \otimes_x y : S = (\Delta \cup \{y : D_1, \ldots, y : D_t\}, \alpha \cup \{(x, y)\})$,

then we change the *(implication) rule* in the obvious way:

from $\Delta, \alpha \vdash^? x : S \Rightarrow B, H$,

step to

$(\Delta, \alpha) \otimes_x y : S \vdash^? y : B, H$,

where S is a set of formulas in NF and $y \notin Lab(\Gamma)$, and we add a rule for proving sets of formulas:

from $(\Delta, \alpha) \vdash^? x : \{B_1, \ldots B_k\}, H$

step to

$(\Delta, \alpha) \vdash^? x : B_i, H$ for $i = 1, \ldots, k$.

Regarding to the simplified formulations without labels, the structural restrictions in the rules (reduction, success) must be globally applied to sets of formulas which are the constituents of databases, considered as a whole; the history H, when is needed, becomes a set of pairs (S_i, A_i), where S_i is a set and A_i is a formula. The property of restricted restart still holds for this formulation.

Example 4.6.2 We show that the following is a theorem of **K5**, using the unlabelled formulation:

$$\square(b \to c) \wedge \square(\square(\square(a \to b) \to d) \to c) \to \square(a \to c).$$

Letting $S_0 = \{b \Rightarrow c, ((a \Rightarrow b) \Rightarrow d) \Rightarrow c\}$, the initial query is

$S_0 \mid \mid \vdash^? \ a \Rightarrow c$

In the derivation below, we make use of restricted restart.

$$\begin{array}{rll}
S_0 \mid \mid & \vdash^? & a \Rightarrow c \\
S_0 \mid \{a\} \mid \{a\} & \vdash^? & c,\ (\{a\}, c) \\
S_0 \mid \{a\} \mid \{a\} & \vdash^? & (a \Rightarrow b) \Rightarrow d, (\{a\}, c) \text{ by reduction wrt. } ((a \Rightarrow b) \Rightarrow d) \Rightarrow c \in S_0 \\
S_0 \mid \{a\}, \{a \Rightarrow b\} \mid \{a \Rightarrow b\} & \vdash^? & d,\ (\{a\}, c) \\
S_0 \mid \{a\}, \{a \Rightarrow b\} \mid \{a\} & \vdash^? & c,\ (\{a\}, c) \text{ by restart} \\
S_0 \mid \{a\}, \{a \Rightarrow b\} \mid \{a\} & \vdash^? & b,\ (\{a\}, c) \text{ by reduction wrt. } b \Rightarrow c \in S_0 \\
S_0 \mid \{a\}, \{a \Rightarrow b\} \mid \{a\} & \vdash^? & a,\ (\{a\}, c) \text{ by reduction wrt. } a \Rightarrow b \\
 & & \text{success.}
\end{array}$$

■

4.6.2 Extension with local clauses

We can extend this language further in a similar way to [Giordano and Martelli 94], by allowing *local clauses* of the form

$G \rightarrow q$,

where $\rightarrow$ denotes ordinary (material) implication. We call them 'local', since $x : G \rightarrow q$ can be used only in world x to reduce the goal $x : q$ and it is not usable/visible in any other world. It is 'private' to x. The extension we suggest may be significant in developing logic programming languages based on modal logic [Giordano et al. 92, Giordano and Martelli 94]. To this purpose, we can introduce a sort of modal Harrop fragment distinguishing database (D-formulas) and goal formulas (G-formulas).

We define below D-formulas i.e. the ones which may occur in the database and G-formulas i.e. the one which may occur in goal position. The former are further distinct in *modal* D-formulas (MD) and *local* D-formulas (LD).

$LD := G \rightarrow q$,
$MD := true \mid q \mid G \Rightarrow MD$,
$D := LD \mid MD$,
$CD := D \mid CD \wedge CD$;
$G := true \mid q \mid G \wedge G \mid G \vee G \mid CD \Rightarrow G$.

We also use $\Box G$ and $\Box D$ as syntactic sugar for $true \Rightarrow G$ and $true \Rightarrow D$. Notice that atoms are both LD- and MD-formulas (as $true \rightarrow q \equiv q$); moreover, any non-atomic MD-formula can be written as $G_1 \Rightarrow \ldots \Rightarrow G_k \Rightarrow q$. Finally, CD formulas are just conjunction of D-formulas. We could have put a more general clause in the definition of MD-formulas, namely $G \Rightarrow D$, rather than the more restrictive $G \Rightarrow MD$. This extension would have allowed

$G_1 \Rightarrow (G_2 \rightarrow q)$,

which is not accepted. But it is easily seen that we have

$G_1 \Rightarrow (G_2 \rightarrow q) \equiv \Box(G_1 \rightarrow (G_2 \rightarrow q)) \equiv G_1 \wedge G_2 \Rightarrow q$.

The last formula is a legal MD-formula, thus the more liberal definition is not really needed.

For D- and G-formulas as defined above we can easily extend the proof-procedure. We give it in the most general formulation for labelled databases. It is clear that one can give an unlabelled formulation for system which allows it, as explained in the previous section. In the labelled formulation, we add the following rules:

- (true) $(\Delta, \alpha) \vdash^? x : true, H$ immediately succeeds.
- (local-reduction) from $(\Delta, \alpha) \vdash^? x : q, H$ step to

  $$(\Delta, \alpha) \vdash^? x : G, H \cup \{(x : q)\}$$

  if $x : G \to q \in \Delta$.
- (and) from $(\Delta, \alpha) \vdash^? x : G_1 \wedge G_2, H$ step to

  $(\Delta, \alpha) \vdash^? x : G_1, H$ and $(\Delta, \alpha) \vdash^? x : G_2, H$.
- (or) from $(\Delta, \alpha) \vdash^? x : G_1 \vee G_2, H$ step

  to $(\Delta, \alpha) \vdash^? x : G_1, H$ or to $(\Delta, \alpha) \vdash^? x : G_2, H$.

Example 4.6.3 Let Δ be the following database

$x_0 : [(\Box p \Rightarrow s) \wedge b] \to q,$
$x_0 : ([(p \Rightarrow q) \wedge \Box a] \Rightarrow r) \to q,$
$x_0 : a \to b.$

We show that $\Delta, \emptyset \vdash^? x_0 : q, \emptyset$ succeeds in the proof system for **KB** and this shows that the formula

$\bigwedge \Delta \to q$ is valid in **KB**.

A derivation is shown in figure 4.5. The property of restricted restart still holds, thus we do not need to record the entire history, but only the first pair (x_0, q). A quick explanation of the steps: step (2) is obtained by local reduction wrt. $x_0 : ([(p \Rightarrow q) \wedge \Box a] \Rightarrow r) \to q$, step (4) by restart, steps (5) and (10) by local reduction wrt. $x_0 : [(\Box p \Rightarrow s) \wedge b] \to q$, step (7) by restart, step (8) by reduction wrt. $x_1 : p \Rightarrow q$ since

$R^{\mathbf{KB}}_{\{(x_0,x_1),(x_0,x_2)\}}(x_1, x_0)$ holds,

step (9) by reduction wrt. $x_2 : \Box p$ since

$R^{\mathbf{KB}}_{\{(x_0,x_1),(x_0,x_2)\}}(x_2, x_0)$ holds,

step (11) by local reduction wrt. $x_0 : a \to b$, step (12) by reduction wrt. $x_1 : \Box a$ since

$R^{\mathbf{KB}}_{\{(x_0,x_1),(x_0,x_2)\}}(x_1, x_0)$ holds.

■

$$\begin{tikzcd}[column sep=small]
(1)\ \Delta \vdash^? x_0 : q, \emptyset \arrow[d, dash] & \\
(2)\ \Delta \vdash^? x_0 : [(p \Rightarrow q) \wedge \Box a] \Rightarrow r, (x_0, q) \arrow[d, dash] & \\
(3)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, \{(x_0, x_1)\} \vdash^? x_1 : r, (x_0, q) \arrow[d, dash] & \\
(4)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, \{(x_0, x_1)\} \vdash^? x_0 : q, (x_0, q) \arrow[d, dash] \arrow[dr, dash] & \\
(5)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, \{(x_0, x_1)\} \vdash^? x_0 : \Box p \Rightarrow s, (x_0, q) \arrow[d, dash] & (10)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, \{(x_0, x_1)\} \vdash^? x_0 : b, (x_0, q) \arrow[d, dash] \\
(6)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, x_2 : \Box p, \{(x_0, x_1), (x_0, x_2)\} \vdash^? x_2 : s, (x_0, q) \arrow[d, dash] & (11)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, \{(x_0, x_1)\} \vdash^? x_0 : a, (x_0, q) \arrow[d, dash] \\
(7)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, x_2 : \Box p, \{(x_0, x_1), (x_0, x_2)\} \vdash^? x_0 : q, (x_0, q) \arrow[d, dash] & (12)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, \{(x_0, x_1)\} \vdash^? x_0 : true, (x_0, q) \\
(8)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, x_2 : \Box p, \{(x_0, x_1), (x_0, x_2)\} \vdash^? x_0 : p, (x_0, q) \arrow[d, dash] & \\
(9)\ \Delta, x_1 : p \Rightarrow q, x_1 : a, x_2 : \Box p, \{(x_0, x_1), (x_0, x_2)\} \vdash^? x_0 : true, (x_0, q) &
\end{tikzcd}$$

Figure 4.5:

The soundness and completeness result of section 4.3 can be extended to this fragment; to this regard let the validity of a query be defined as in that section.

Theorem 4.6.4 $\Delta \vdash^? G, H$ *succeeds in* $P(\mathbf{S})$ *if and oly if it is valid.*

Proof.(Sketch) The soundness part is proved by induction on the height of derivations, we omit the details.

In order to prove the completeness we need the following properties:

1. Suppose that

   (a) $\Gamma \cup \{u : G \rightarrow q\}, \alpha \vdash^? y : G_1, H$ succeeds and
   (b) $\Gamma, \alpha \vdash^? u : G, H$ succeeds implies $\Gamma, \alpha \vdash^? u : q, H$ succeeds,

   then $\Gamma, \alpha \vdash^? y : G_1, H$ succeeds.

2. Suppose that

   (a) $\Gamma \cup \{u : G_1 \Rightarrow \ldots \Rightarrow G_k \Rightarrow q\}, \alpha \vdash^? y : G_1, H$ succeeds and
   (b) for every $u = x_0, x_1, \ldots, x_k$, if $A^{\mathbf{S}}_{\alpha}(x_{i-1}, x_i)$ and $\Gamma, \alpha \vdash^? x_i : G_i$ succeeds for $i = 1, \ldots, k$ then also $\Gamma, \alpha \vdash^? x_k : q, H$ succeeds,

   then $\Gamma, \alpha \vdash^? y : G_1, H$ succeeds.

The proof of this properties is by a simple induction on the lenght of a derivation of hypothesis (a) in both cases.

Then the completeness part is proved by contrapposition. If a query does not succeed, we build a model in which it is not valid; the construction is the same as the one in theorem 4.3.4. To this regard, we start with an enumeration with infinite repetitions of pairs of the form (x_i, G_i) where G_i are G formulas which are neither conjunctions, nor disjunctions. The construction then proceeds exactly as in theorem 4.3.4 leading to the definition of a canonical model $M = (W, R, V)$. We then prove by mutual induction the following claims:

- (i) for all $x \in W$ and G-formulas G, $M, x \models G \;\Leftrightarrow\; \exists n\; x \in Lab(\Gamma_n) \wedge \Gamma_n, \alpha_n \;\vdash^{?}\; x : G, H_n$ succeeds.
- (ii) for all $x \in W$ and D-formulas D, if $D \in \Delta_n$ then $M, x \models D$.

From these facts the theorem follows immediately. □

Further extensions might be interesting: one may think of taking the computation for classical logic we have seen in chapter 2 and combine it with the modal procedures of this chapter. By means of a simple translation the resulting procedure could handle the full propositional modal logic. We conjecture that the procedure is complete, although we have not checked it. In any case, handling formulas of the form $\Diamond A$ by translating them into $\Box(A \rightarrow \bot) \rightarrow \bot$, (or equivalently into $(A \Rightarrow \bot) \rightarrow \bot$), is probably not be the most effective way of extending the goal-directed computation.

4.7 A Further case study: modal Logic G

In this section we give a goal directed procedure for the implicational fragment of modal logic **G**. Modal logic **G** was originally introduced as a modal interpretation of the notion of formal provability in Peano Arithmetic (see [Boolos 79]).

Semantically, **G**is characterized by the following conditions on the accessibility relation R: (i) R is transitive, and (ii) R does not have infinitely increasing chains $w_0Rw_1, \ldots, w_iRw_{i+1}, \ldots$.

To deal with G we adopt our deduction system for **K4** and we modify the rule for implication goal in order to take care of the finiteness condition. To explain the idea intuitively, suppose we want to show that $A \Rightarrow B$ holds in a world w, we can argue by reduction ab absurdum as follows: assume that $A \Rightarrow B$ is false at w. Then there is a world w' such that wRw', in which $A \wedge \neg B$ is true. By the finiteness condition, we can assume that there is a *last* world $w*$ in which $A \wedge \neg B$ is true. In $w*$, A holds, B does not hold, but also $A \Rightarrow B$ holds, since every world w'' accessible from $w*$ will not satisfy $A \wedge \neg B$. We have then to show that B cannot be false in $w*$.

According to the above argument, we can modify the rule for implication in the following way: to evaluate a goal $A \Rightarrow B$, from a database Δ, we add to Δ the formula $A \wedge (A \Rightarrow B)$, and we ask B [5]. We show that this rule is sound and complete for **G**.

We work with the unlabelled formulation of the implicational fragment of **K4**, and we further notice that we do not need to introduce conjunction, but only to take *pairs* of formulas as the unit elements of databases. A database Δ is a sequence of pairs of formulas:

$S_1, \ldots, S_i,$

[5]Our modified implication rule closely corresponds to the rule for necessity in **G**, within tableau formulation: if $\neg\Box A$ is in a branch then create a new world with $\neg A$ and $\Box A$. [Fitting 83]

where each S_i is a pair of formulas (A_i, B_i). For implication we have the following:

- (implication) from $\Delta \vdash^? A \Rightarrow B$ step to $\Delta, (A, A \Rightarrow B) \vdash^? B$.

The other two rules reduction and success, are the same as in the unlabelled version of P(**K4**), that is **K4** with conjunction, whose rules we review below for clarity:

- (success) $S_1, \ldots, S_n \vdash^? q$ succeeds if $q \in S_n$.
- (reduction) From $S_1, \ldots, S_n \vdash^? q$ step to

  $S_1, \ldots, S_{j_i} \vdash^? D_i$, for $i = 1, \ldots, k$

  if there is a formula $A = D_1 \Rightarrow \ldots \Rightarrow D_k \Rightarrow q \in S_j$, for some j, and integers $j < j_1 < \ldots < j_k = n$.

Example 4.7.1 We show a derivation of

$$F = ((b \Rightarrow a) \Rightarrow a) \Rightarrow b \Rightarrow a$$

which is equivalent (in K) to Löb's axiom: $\Box(\Box A \to A) \to \Box A$

$$\begin{array}{rcll}
 & \vdash^? & ((b \Rightarrow a) \Rightarrow a) \Rightarrow b \Rightarrow a & (4.1)\\
((b \Rightarrow a) \Rightarrow a, F) & \vdash^? & b \Rightarrow a & (4.2)\\
((b \Rightarrow a) \Rightarrow a, F), (b, b \Rightarrow a) & \vdash^? & a & (4.3)\\
((b \Rightarrow a) \Rightarrow a, F), (b, b \Rightarrow a) & \vdash^? & b \Rightarrow a & (4.4)\\
((b \Rightarrow a) \Rightarrow a, F), (b, b \Rightarrow a), (b, b \Rightarrow a) & \vdash^? & a & (4.5)\\
((b \Rightarrow a) \Rightarrow a, F), (b, b \Rightarrow a), (b, b \Rightarrow a) & \vdash^? & b & (4.6)\\
 & & \textit{success} & (4.7)
\end{array}$$

On step (3) a can match only with $(b \Rightarrow a) \Rightarrow a$; on step (5) a can match with $b \Rightarrow a$ in the second pair from the left. ■

Proposition 4.7.2 *Let $A_1, \ldots, A_n$, $B_1, \ldots, B_n$, C be any formulas; if $(A_1, B_1), \ldots, (A_n, B_n) \vdash^? C$ succeeds, then the formula $(A_1 \wedge B_1) \Rightarrow (A_2 \wedge B_2) \ldots \Rightarrow (A_n \wedge B_n) \Rightarrow C$ is valid in* **G**. *In particular, if $\emptyset \vdash^? A$ succeeds, then A is valid in G.*

Proof. The proof of the theorem is based on the following fact: for every A and B:

$$\models_{\mathbf{G}} A \Rightarrow B \leftrightarrow (A \wedge (A \Rightarrow B)) \Rightarrow B.$$

To see this, we have

$$\begin{array}{l}
\models_{\mathbf{G}} A \Rightarrow B \leftrightarrow \Box(A \to B)\\
\models_{\mathbf{G}} A \Rightarrow B \leftrightarrow \Box(\Box(A \to B) \to (A \to B))\\
\models_{\mathbf{G}} A \Rightarrow B \leftrightarrow \Box((A \wedge \Box(A \to B)) \to B)\\
\models_{\mathbf{G}} A \Rightarrow B \leftrightarrow (A \wedge (A \Rightarrow B)) \Rightarrow B
\end{array}$$

Then, the proof of the proposition proceeds by induction on the height of a successful derivation, using the previous fact when A is an implication. □

In order to prove completeness we make use of the translation of **G** in **K4** proposed by Balbiani and Herzig [Balbiani and Herzig 94], that we adapt to the language $\mathcal{L}(\Rightarrow)$, and then we can rely on the completeness of our proof system for **K4**. The translation is the following:

$p^+ = p^- = p$, for any atom p,
$(A \Rightarrow B)^+ = (A^- \wedge (A \Rightarrow B)) \Rightarrow B^+)$
$(A \Rightarrow B)^- = (A^+ \Rightarrow B^-)$.

Notice that the translation of $(A \Rightarrow B)^+$ clearly resembles the way we deal with strict implication of **G**. In [Balbiani and Herzig 94] it is proved the following fact.

Proposition 4.7.3 ([Balbiani and Herzig 94]) *For any formula A, A is a theorem of* **G** *iff A^+ is a theorem of* **K4**.

Lemma 4.7.4 *For all databases Δ, Π, sets S, and formulas H in $\mathcal{L}(\Rightarrow, \wedge)$, for all formulas $A \in \mathcal{L}(\Rightarrow)$ the following are true:*

1. *if $\Delta, S \cup \{A^+\}, \Pi \vdash^? H$ succeeds in P(***K4***), then also $\Delta, S \cup \{A\}, \Pi \vdash^? H$ succeeds in P(***K4***);*
2. *if $\Delta, S \cup \{A\}, \Pi \vdash^? H$ succeeds in P(***K4***), then also $\Delta, S \cup \{A^-\}, \Pi \vdash^? H$ succeeds in P(***K4***);*
3. *if $\Delta \vdash^? A$ succeeds in P(***K4***), then also $\Delta \vdash^? A^+$ succeeds in P(***K4***);*
4. *if $\Delta \vdash^? A^-$ succeeds in P(***K4***), then also $\Delta \vdash^? A$ succeeds in P(***K4***).*

Moreover, if the query in the hypothesis of each claim (1) - (4) has a successful derivation of height h, then the query in the thesis has a successful derivation of height no more than h.

Proof. Claims (1) - (4) are proved by a simultaneous induction on pairs (c, h) lexicographically ordered, where $c = cp(A)$, and h is the height of a derivation of the query in the hypothesis of each (1) - (4). We omit the details. □

Now we can easily prove the completeness of P(**G**). We show that the deduction rules for **G** compute, so to say, the *run time* translation of a query in **K4**.

Lemma 4.7.5 *for any formulas $A_1, \ldots, A_n$, $B_1, \ldots, B_n$, C of the language $\mathcal{L}(\Rightarrow)$, the following holds: if $\{A_1^-, B_1\}, \ldots, \{A_n^-, B_n\} \vdash^? C^+$ succeeds in P(***K4***), then $(A_1, B_1), \ldots, (A_n, B_n) \vdash^? C$ succeeds in P(***G***). In particular, if $\vdash^? A^+$ succeeds in P(***K4***), then $\vdash^? A$ succeeds in P(***G***).*

Proof. By induction on the height h of a succeeding derivation of the query in the hypothesis.

- $h = 0$, then C must be an atom, so that $C^+ = C$ and either $C = A_n^+ = A_n$, or $C = B_n$, in both cases the result follows immediately.
- $c > 0$, we distinguish two cases: $C = X \Rightarrow Y$, or C is an atom. In the former case we have: if $\{A_1^-, B_1\}, \ldots, \{A_n^-, B_n\} \vdash^? C^+$ succeeds (in P(**K4**)) with height h, $C^+ = \{X^-, X \Rightarrow Y\} \Rightarrow Y^+$, we have that

  $$\{A_1^-, B_1\}, \ldots, \{A_n^-, B_n\}, \{X^-, X \Rightarrow Y\} \vdash^? Y^+ \text{ succeeds (in P(}\mathbf{K4}\text{)) with height } h-1;$$

  thus by inductive hypothesis, we have that

$(A_1, B_1), \ldots, (A_n, B_n), (X, X \Rightarrow Y) \vdash^? Y$ succeeds in P(**G**),

and therefore also $(A_1, B_1), \ldots, (A_n, B_n) \vdash^? X \Rightarrow Y$ succeeds in P(**G**).

Suppose now that C is an atom q, we have two subcases: (a) q unifies with a formula A_i^-, or (b) q unifies with a formula B_i. In case (a) let $A_i = D_1 \Rightarrow \ldots D_k \Rightarrow q$, then A_i^- is $D_1^+ \Rightarrow \ldots D_k^+ \Rightarrow q$. Then there are $j_1, \ldots, j_k$, with $i < j_1 < j_2 < \ldots < j_k = n$, such that for $l = 1, \ldots, k$

$$\{A_1^-, B_1\}, \ldots, \{A_{j_l}^-, B_{j_l}\} \vdash^? D_l^+$$

succeeds in P(**K4**) with height $h_l < h$. By inductive hypothesis, we may conclude that

$(A_1, B_1), \ldots, (A_{j_l}, B_{j_l}) \vdash^? D_l$ succeeds in $P(G)$, for $l = 1, \ldots, k$,

so that also

$(A_1, B_1), \ldots, (A_n, B_n) \vdash^? q$ succeeds in $P(G)$.

In case (b) let $B_i = E_1 \Rightarrow \ldots E_z \Rightarrow q$. Then there are $j_1, \ldots, j_z$, with $i < j_1 < j_2 < \ldots < j_z = n$, such that for $l = 1, \ldots, z$ $\{A_1^-, B_1\}, \ldots, \{A_{j_l}^-, B_{j_l}\} \vdash^? E_l$ succeeds in P(**K4**) with height $h_l < h$. By the previous lemma, we have that also

$\{A_1^-, B_1\}, \ldots, \{A_{j_l}^-, B_{j_l}\} \vdash^? E_l^+$ succeeds in P(**K4**) with height $h_l' \leq h_l < h$.

We may apply the inductive hypothesis, and conclude that $(A_1, B_1), \ldots, (A_{j_l}, B_{j_l}) \vdash^? E_l$ succeeds in $P(G)$, for $l = 1, \ldots, z$, so that also $(A_1, B_1), \ldots, (A_n, B_n) \vdash^? q$ succeeds in P(**G**).

□

Theorem 4.7.6 *If A is valid in* **G**, *then $\emptyset \vdash^? A$ succeeds in P(*__G__*).*

Proof. Let $A = X_1 \Rightarrow X_2 \Rightarrow \ldots \Rightarrow X_n \Rightarrow q$. We have that

$$A^+ = \{X_1^-, F_1\} \Rightarrow \{X_2^-, F_2\} \Rightarrow \ldots \{X_n^-, F_n\} \Rightarrow q,$$

where $F_n = X_n \Rightarrow q$, $\quad F_i = X_i \Rightarrow F_{i+1}$. If A is a theorem of **G**, we have that A^+ is a theorem of **K4**. Thus, $\emptyset \vdash^? A^+$ succeeds in P(**K4**). From lemma 4.7.5 we finally get that $\emptyset \vdash^? A$ succeeds in the proof system P(**G**). □

4.8 Extension to Horn Modal logics

The goal-directed proof procedure can be easily generalized to a broader class of logics. We call *Horn modal logics* the class of modal logics which are semantically characterized by Kripke models in which the accessibility relation is definable by means of Horn conditions. This class has been studied in [Basin et al. 97a]. All previous examples, with the exception of Gödel logic **G**, fall under this class. To show how to obtain this generalization, we have to change a bit the presentation of the proof procedure. In the presentation we have given, a database is a set of labelled formulas together with a set of links (pairs of labels) α. For each specific system, we have introduced predicates $A_\alpha^{\mathbf{S}}$ which specify closure conditions (reflexive, transitive, etc.) of the relation α. This predicates are external to the language and we have supposed to have an external mechanism which allows us to prove statements of the form $A_\alpha(x, y)$. Following [Basin et al. 97a], we now define a database as a pair $\Delta = \langle \Delta_F, \Delta_R \rangle$, where Δ_F is a set of labelled formulas as before and Δ_R is a set of Horn formulas in the language $\mathcal{L}(\mathcal{A}, R)$. A Horn formula has the form

$$\forall(R(t_1,t'_1)\wedge\ldots\wedge R(t_n,t'_n)\rightarrow R(t,t'))$$

where each t_i ie either a variable X_i, or a constant $x_i\in\mathcal{A}$; the notation $\forall$ denotes the universal closure, that we will omit henceforth [6]. The idea is clearly that the formulas of Δ_R define the accessibility relation. By

$$\Delta_R\vdash R(t_1,t_2).$$

we denote standard provability in classical logic for Horn clauses. We rephrase the reduction and implication rule as follows. Let $\Delta=\langle\Delta_F,\Delta_R\rangle$.

- (implication) from $\langle\Delta_F,\Delta_R\rangle\ \vdash^?\ x:A\Rightarrow B,H$, step to

  $$\langle\Delta_F\cup\{y:A\},\Delta_R\cup\{R(x,y)\}\rangle\ \vdash^?\ y:B,H,$$

  where $y\in\mathcal{A}\wedge y\notin Lab(\Delta)$.

- (reduction) if $y:C\in\Delta_F$, with $C=B_1\Rightarrow B_2\Rightarrow\ldots\Rightarrow B_k\Rightarrow q$, with q atomic, then from

  $$\langle\Delta_F,\Delta_R\rangle\ \vdash^?\ x:q,H$$

  step to

  $$\langle\Delta_F,\Delta_R\rangle\ \vdash^?\ u_1:B_1,H\cup\{(x,q)\},\quad\ldots,\quad\langle\Delta_F,\Delta_R\rangle\ \vdash^?\ u_k:B_k,H\cup\{(x,q)\},$$

  for some $u_0,\ldots,u_k\in Lab(\Delta)$, with $u_0=y$, $u_k=x$, such that

  for $i=0,\ldots,k-1$, $\Delta_R\vdash R(u_i,u_{i+1})$ holds.

Soundness and completeness can be easily extended to the class of Horn modal logics. Completeness relies on the cut-admissibility property which holds also in this case. To this concerning, let us define, given $\Delta=\langle\Delta_F,\Delta_R\rangle$

$$A_{\Delta_R}=\{(x,y)\mid x,y\in\mathcal{A}\ \wedge\ \Delta_R\vdash R(x,y)\}.$$

It is almost trivial to see that A_{Δ_R} satisfies the conditions (i)-(iii) of theorem 4.2.3 reformulated in this new setting, i.e.

(i) $A_{\Delta_R}[u/v]=A_{\Delta_R[u/v]}$, where $A_{\Delta_R}[u/v]$ is the image of A_{Δ_R} under the mapping $\phi(u)=v$ and $\phi(x)=x$, for $x\neq u$;
(ii) $A_{\Delta_R}\subseteq A_{\Delta_R\cup\Delta'_R}$,
(iii) $(x,y)\in A_{\Delta_R}$, then $A_{\Delta_R\cup\{R(x,y)\}}=A_{\Delta_R}$.

Notice that the above conditions hold even if Δ_R is an arbitrary set of first-order formulas containing the relational symbol R. The reason why the restriction to Horn matters is that A_{Δ_R} as defined above determines *one* model of Δ_R. More precisely, let us define a first-order structure M_{Δ_R} for a language which includes the relational symbol R (namely a Herbrand model), with

$$M_{\Delta_R}(R)=A_{\Delta_R},$$

[6]We can introduce a further distinction between an extensional part of Δ_R corresponding to the old α and an intensional part containing the Horn formulas; the former varies for each database, whereas the latter is fixed for each modal logic.

then we have that $M_{\Delta_R} \models \Delta_R$. This means that the relation A_{Δ_R} satisfies the properties of each modal systems (which are expressed by the non-atomic non-closed Horn formulas in Δ_R). In general, this fact does not hold unless Δ_R is equivalent to a set of Horn formulas. To see this, consider the property of linearity

$$L \equiv \forall XY(R(X,Y) \vee R(Y,X)).$$

Let $\Delta_R = \{L, R(u_1,u_2), R(u_1,u_3)\}$, then

$$\Delta_R \vdash R(u_2,u_3) \vee R(u_3,u_2), \text{ but } \Delta_R \not\vdash R(u_2,u_3) \text{ and } \Delta_R \not\vdash R(u_3,u_2).$$

If in the database we have $u_2 : A \Rightarrow q$ and $u_3 : A$ and $R(u_2,u_3)$ holds, we can conclude $u_3 : q$, otherwise we cannot. In other words, we must embody in the computation some form of case-analysis to make it work. Notice that in this case $A_{\Delta_R} = \emptyset$ and clearly $M_{\Delta_R} \not\models \Delta_R$.

Let us go back to the completeness for Horn modal logics. If we inspect the completeness proof of (theorem 4.3.4), we only need to check one condition, namely that the properties of the accessibility relation are preserved under countable unions of chains of relations. Given a sequence of databases $\Delta_i = \langle \Delta_{F,i}, \Delta_{R,i}, i \in \omega\rangle$ such that $\Delta_i \subseteq \Delta_{i+1}$, we define

$$A_R = \bigcup_{i\in\omega} A_{\Delta_{R,i}} \quad \Delta_R = \bigcup_{i\in\omega} \Delta_{R,i},$$

and a structure M_R with $M(R) = A_R$. Then we can easily prove that $M_R \models \Delta_R$. That is to say, the limit (or union) of $A_{\Delta_{R,i}}$ satisfies the properties of each modal system. This fact is not ensured unless each Δ_R is Horn.

IS IT RIGHT OR IT HOLDS FOR A MUCH BIGGER CLASS??

4.9 Related work

Many authors have developed sequent calculi for modal logics, (see seminal works by Fitting [Fitting 83], and [Gore 99] for a recent survey). We cannot give a full account of the research in this area. We limit our consideration to proof systems which have some relation with ours.

Cerrato in [Cerrato 94] gives a Fitch-style natural deduction formulation of the 15 basic modal systems, where strict implication is the main connective. The central notion is that one of *strict subproof*, which is a proof of a formula whose main connective is strict implication. Strict implication represents the deducibility link between the hypothesis and the conclusion of a strict subproof. The elimination rule for strict implication in the basic system **K**is as follows: if $A \Rightarrow B$ occur in the parent proof, one can import $A \rightarrow B$ (material implication) in any immediate strict subproof. Stronger systems are obtained by allowing *categorical strict subproof* (system **T**), and by adding some suitable reiteration rules (for importing a formula from a proof to its strict subproofs) and contrapposition rules (systems with **D** and **B**). For instance, in **K4** one can reiterate any strict implication formula from a proof to its strict subproofs. In case of **K**, to give an example, a proof of $A_1, \ldots, A_n \vdash^? B$ in our calculus seems to correspond to a natural deduction proof of B in the n-times nested subproof with strict hypothesis A_n, (the nested strict subproof at level i has hypothesis A_i). A more precise mapping is still to be investigated.

Masini [Masini92] develops a cut-free calculus for modal logic **KD** based on 2-dimensional sequents, each side of a sequent is a vertical succession of sequences of formulas (1-sequences), such a

$$
\frac{\begin{array}{lcl}\Gamma & & \\ \alpha, A & \vdash & B\end{array}}{\begin{array}{lcl}\Gamma & & \\ \alpha & \vdash & A \Rightarrow B\end{array}}
\qquad
\frac{\begin{array}{lcl}\Gamma & & \\ \sigma & & \\ \alpha & \vdash & A\end{array}\qquad\begin{array}{lcl}\Gamma & & \\ \sigma & & \\ \alpha, B & & \\ \Gamma' & \vdash & C\end{array}}{\begin{array}{lcl}\Gamma & & \\ \sigma, A \Rightarrow B & & \\ \alpha & & \\ \Gamma' & \vdash & C\end{array}}
$$

Figure 4.6:

vertical sequence is called a 2-sequence; finally, a sequent is a pair of two 2-sequences; we refer to [Masini92] for further terminology. Modal rules act on different levels of the 2-sequences, whereas propositional rules act on formulas on the same level of the 2-sequences which constitute a sequent. Masini also presents an intuitionistic version of his calculus (for logic KD). There is some connection with our systems. If we restrict our consideration to the language $K(\Rightarrow, \wedge)$, an unlabelled query of the form $S_1, \ldots, S_n \vdash^? A$ corresponds to an (intuitionistic) 2-sequent of the form

$$
\begin{array}{lcl}
S_1 & & \\
\vdots & \vdash & \epsilon^{n-1} \\
S_n & & A
\end{array}
$$

Namely, one can develop a 2-sequent calculus for $K(\Rightarrow, \wedge)$, containing, for strict implication, the rules of Fig.4.6 [7]. Axioms and the rules for conjunction are as in Masini's formulation. It is not difficult to prove that (a) the above calculus is sound, and (b) every successful derivation of a query $S_1, \ldots, S_n \vdash^? A$ in our proof system $K(\Rightarrow, \wedge)$ can be mapped into a (cut-free) derivation of

$$
\begin{array}{lcl}
S_1 & & \\
\vdots & & \\
S_n & \vdash & A
\end{array}
$$

in the above calculus. Thus, the calculus is also complete. Although Masini has not developed calculi for other modal systems, and for strict implication language, we conjecture that the above considerations (2-sequent calculus and the mapping with our proof system) can be extended to some other modal systems.

Giordano, Martelli and Rossi [Giordano et al. 92], [Giordano and Martelli 94] have developed goal-directed methods for fragments of first-order (multi-)modal logics. Their extension is motivated by several purposes: introducing scoping constructs (such as blocks and modules) in logic programming, representing epistemic and inheritance reasoning. The spirit of their work is very close to the material presented in this chapter. In particular in [Giordano and Martelli 94] they have defined a family of

[7] The following calculus makes another simplification wrt. the original formulation, the only formula in the consequent is always at the same level as maximum level of the 2- sequence in the antecedent, thus we can omit ϵ^k in the consequent.

first-order logic programming languages, based on modal logic **S4** to represent a variety of scoping mechanism. If we restrict our consideration to the propositional level, their languages are strongly related to the one defined in section 4.6.2 in the case the underlying logic is **S4**. The richer (propositional) fragment of **S4**they consider is the language L_4 which contains

$$G := \mathit{true} \mid \Box q \mid G \wedge G \mid \Box(D \rightarrow G),$$
$$D := \Box(G \rightarrow \Box q) \mid G \rightarrow \Box q \mid D \wedge D,$$

where q ranges on atoms, D on database formulas (D-formulas), and G on goal formulas (G-formulas). This language is very close to the one defined in section 4.6.2, although neither one of the two in contained in the other. The (small) difference is that in L_4 D-formulas of the form $G \rightarrow \Box q$ are allowed, and they are not in the language of section 4.6.2; on the other hand D-formulas of the form $\Box(G_1 \rightarrow \Box(G_2 \rightarrow q))$ are allowed in the latter, but not in L_4. Of course it would not be difficult to join the languages. The proof procedure they give (at the propositional level) is essentially the same as the unlabelled version of P(**S4**) we have seen in section 4.6.2.

Basin, Mattews and Vigano have developed a proof-theory for modal logics, based on the labelled mechanism [Basin et al. 97a, Basin et al. 99]. A much related approach, along somewhat different lines was presented in [Gabbay 96] and [Russo 96]. The authors have developed both sequent and natural deduction systems for several modal logics which are completely uniform. Database are distinguished into a relational part Δ_R and into a logical part Δ_F as explained in section 4.8. The class of modal logics they consider is larger than the one considered in this chapter (with the exception of **G**), they are able to develop proof system for full modal logics including the case of seriality and convergency. In further work [Basin et al. 97b], they have studied labelled sequent calculi to obtain space-complexity bounds for modal logics. The relation between our and their work might be established by giving sequent calculi for strict implication fragment as we have exemplified above for Masini's formulation. Once we had defined labelled sequent calculi based on strict implication, one might wonder if our goal-directed proofs correspond to *uniform proof* (according to the definition by Miller [Miller et al. 91]) in these calculi. We think so, although we have not investigated this issue further.

SHALL WE PUT WHAT FOLLOWS???

The use of labels to represent worlds for modal logics is rather old and goes back to Fitting. In the seminal work [Fitting 83] formulas are labelled by string of atomic labels (world prefixes), which represents paths of accessible worlds. The rules for modalities are the same for every system: for instance if a branch contain $\sigma : \Box A$, and σ' is accessible from σ, then one can add $\sigma' : A$ in the same branch. For each system, there are some specific accessibility conditions on prefixes which constraint the propagation of modal formulas.

Wansing [Wansing 90] develops sequent calculi in the framework of Belnap's display logic for the basic 15 systems. Sequents are pair of structures generated from formulas by means of some operators (namely, I, $\circ$, $\bullet$, $*$, together with the boolean constants). This framework is very general and uniform and, in particular, the rules for modalities are the same for all systems. To obtain the minimal system K, a certain number of structural rules have to be postulated; the other systems are obtained by postulating additional structural rules.

Chapter 5

Substructural Logics

5.1 Introduction

In this chapter we consider substructural implicational logics. These logics, as far as the implicational fragment is concerned, put some restrictions on the use of formulas in deductions. The restrictions may require either that every formula of the database must be used, or that it cannot be used more than once, or that it must be used according to a given ordering of database formulas. The denomination *substructural logics* can be explained by the fact that, these logical systems restrict (some of) the structural rules of deduction, that is to say the *contraction*, *weakening* and *permutation* rules which are allowed both by classical and intuitionistic provability. Substructural logics have received an increasing interest in computer science community because of their potential applications in a number of different areas, such as natural language processing, database update, logic programming, type-theoretic analysis by means of the so called Curry-Howard isomorphism, and, more generally the cathegorical interpretation of logics. We refer to for a survey [**?**]. Recently, Routley-Meyer semantics for substructural logics has been re-interpreted as modelling agent-interaction [].

We give a brief survey of the implicational systems we consider in this chapter. The background motivation of *relevance logics* is the attempt to formalize an intuitive requirement on logical deductions: in any deduction the hypotheses must be relevant to the conclusion. *Relevant* means that the hypotheses must be actually used to get the conclusion. The first intuition behind relevance logics is thus to reject the *weakening rule*:

$$\frac{\Gamma \vdash A}{\Gamma, B \vdash A,}$$

since B may have nothing to do with the proof of A from Γ. In the object language this rule is reflected by the theorem (of classical, likewise intuitionistic logic)

(Irrelevance) $A \to (B \to A)$.

The implicational fragment of **R**can be axiomatized by dropping Irrelevance from the axiomatization of intuitionistic implication. A sequent calculus for the implicational fragment of **R** can be easily obtained from the sequent formulation of (the implicational fragment of) intuitionistic logic, by restricting the identity axiom to single formulas $A \vdash A$, and by dropping the above weakening rule. Whereas system **R** takes into account the basic concern about use, other systems such as **E** and **T** put some further restriction on the order in which formulas can be used in any deduction. These restriction can be expressed in a sequent formulation as restrictions on the rule of *permutation* (see below), although in a less natural way[1]. None of these systems put any control on the number of times in which a formula can be used within a deduction, what only matters is that it is used at least once.

On the other hand, a few systems we consider pay attention to the number of times a formula is used in a deduction. In some logics, such as **BCK**and **L**, a formula cannot be used more than once. From the point of view of a sequent system formulation, this constraint about the use of a formula, can be expressed by removing the rule of contraction:

$$\frac{\Gamma, A, A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B} \ ,$$

from the standard sequent system for intuitionistic logic. In this context, formulas may be thought as representing resources which are consumed through a logical deduction. This motivates an alternative denomination of substructural logics as *bounded resource logics* []. In more detail, **BCK**is the logic which result from intuitionistic logic by dropping contraction, **L**(linear logic) reject both weakening and contraction [2].

The weakest system we consider is **CL**, which corresponds to the *right implicational* fragment of (associative ???) Lambek calculus, proposed by Lambek in 1958 [**?**] as a calculus of syntactic categories in natural language. This system reject all substructural rules, in particular, in addition to weakening and contraction, it rejects the law of permutation:

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C,}$$

As we have seen in the Introduction, we ccan ontrol the use of formulas by labelling data and putting constraints on the labels. In this specific context by labelling data, we are able to record whether they have been used or not and to express the additional conditions needed for each specific systems. The use of labels to deal with substructural logics is not a novelty, it has been introduced by Anderson and Belnap [Anderson and Belnap 75] to develop a natural-deduction formulation of most relevance logics. Our proof systems are similar to their natural deduction systems in this respect: we do not have explicit structural rules. The structural rules are *internalized* as resctrictions in the logical rules.

5.2 Proof Systems

Definition 5.2.1 Let us fix a denumerable alphabet $\mathcal{A} = \{x_1, \ldots, x_i, \ldots\}$ of labels. We assume that labels are totally ordered as shown in the enumeration, $\mathbf{v_0}$ is the first label. A *database* is a finite set of labelled formulas $\Delta = \{x_1 : A_1, \ldots, x_n : A_n\}$. We assume that

if $x : A \in \Delta, x : B \in \Delta$, then $A = B$ [3].

We use the notation $Lab(E)$ for the set of labels occurring in an expression E, and we finally assume that $\mathbf{v_0} \notin Lab(\Delta)$. Label $\mathbf{v_0}$ will be used for queries from the empty database.

■

[1]Belnap, Wansing, Dozen (rigth quotations) [] have extensively studied sequent calculi for substructural logics. They propose to structure sequent by non-associative operators in the style of Belnap Display logic.

[2]In linear logic, these rules are re-introduced in a controlled way by special operators called exponentials.

[3]This restriction will be lifted in section 5.1 where conjunction is introduced in the language.

Definition 5.2.2 A query Q is an expression of the form:

$$\Delta, \delta \vdash^? x : G$$

where Δ, is a database, δ is a finite set of labels not containing $\mathbf{v_0}$; moreover if $x \neq \mathbf{v_0}$, then $x \in Lab(\Delta)$, and G is a formula.

A query from the empty database has the form:

$$\vdash^? \mathbf{v_0} : G.$$

■

By convention, we stipulate that if $\delta = \emptyset$, then $max(\delta) = \mathbf{v_0}$. The set of labels δ may be thought as denoting the set of resources that are available to prove the goal. Label x in front of the goal has a double role as "a position" in the database from which the goal is asked and a as available resource.

5.2.1 Goal-directed proof procedures

The rules for success and reduction are parametrized to some conditions $Succ^S$ and Red^S which will be defined below. Here below are the deduction rules for a query of the form:

$$\Delta, \delta \vdash^? x : G.$$

- (success)

  $$\Delta, \delta \vdash^? x : q; \text{ succeeds,}$$

  if $x : q \in \Delta$ and $Succ^S(\delta, x)$.

- (implication) from

  $$\Delta, \delta \vdash^? x : C \to G$$

  we step to

  $$\Delta \cup \{y : C\}, \delta \cup \{y\} \vdash^? y : G,$$

  where $y > max(Lab(\Delta))$, (hence $y \notin Lab(\Delta)$);

- (reduction) from

  $$\Delta, \delta \vdash^? x : q,$$

  if there is some $z : C \in \Delta$, with

  $$C : A_1 \to \ldots \to A_k \to q,$$

  and there are δ_i, and x_i for $i = 0, \ldots, k$ such that:

  1. $\delta_0 = \{z\}$, $x_0 = z$,
  2. $\bigcup_{i=0}^{k} \delta_i = \delta$.
  3. $Red^S(\delta_0, \ldots, \delta_k, x_0, \ldots, x_k; x)$

  then for $i = 1, \ldots k$, we step to

  $$\Delta, \delta_i, \vdash^? x_i : A_i.$$

The conditions for success are either (s1) or (s2) according to each system:

Condition	(r0)	(r1)	(r2)	(r3)	(r4)	(Success)
CL	*				*	(s2)
T-W	*	*		*		(s2)
T	*			*		(s2)
E-W	*	*	*			(s2)
E	*		*			(s2)
L		*				(s2)
R						(s2)
BCK		*				(s1)

- (s1) $Succ^S(\delta, x) \;\equiv\; x \in \delta$,
- (s2) $Succ^S(\delta, x) \;\equiv\; \delta = \{x\}$.

The conditions Red^S are obtained as combination of the following clauses:

- (r0) $x_k = x$;
- (r1) for $i, j = 0, \ldots, k$, $\delta_i \cap \delta_j = \emptyset$;
- (r2) for $i = 1, \ldots, k$, $x_{i-1} \leq x_i$ and $max(\delta_i) \leq x_i$;
- (r3) for $i = 1, \ldots, k$, $x_{i-1} \leq x_i$ and $max(\delta_i) = x_i$;
- (r4) for $i = 1, \ldots, k$, $x_{i-1} < x_i$, $max(\delta_{i-1}) = x_{i-1} < min(\delta_i)$ and $max(\delta_k) = x_k$.

The conditions Red^S are then defined according to the table in figure 5.2.1

Notice that

$(r4) \Rightarrow (r3) \Rightarrow (r2)$, and
$(r4) \Rightarrow (r1)$.

We give a quick explanation of the conditions $Succ^S$ and Red^S. Remember that δ are resources which must/can be used.

For success rule, in all cases but **BCK**, we have that we can succeed if $x : q$ is in the database, x is the only resource left, and q is asked from position x; in case of **BCK**, x must be among the available resources, but we do not require that x is the only one left.

The conditions for reduction can be explained as follows: resources δ are split in several δ_i, for $i = 1, \ldots, k$ and each part δ_i must be used in a derivation of a subgoal A_i.

In case of logics *without contraction* we cannot use a resource twice, that is why, by restriction (r1), the δ_i's must be disjointed and z, the label of the formula we are using in the reduction step, is no longer available.

Restriction (r2) impose that successive subgoals are to be proved from successive positions in the database: only positions $y \geq x$ are "accessible" from x; moreover each x_i must be accessible from resources in δ_i. Notice that the last subgoal A_k must be proved from x, the position from which the atomic goal q is asked.

Restriction (r3) is similar to (r4) but it further requires that the position x_i is among the available resources δ_i.

Restriction (r4) forces the goal A_i to be proved by using successive disjointed segments δ_i of δ. Moreover, z which labels the formula we are using in reduction, must be the first resource among the available ones.

It is not difficult to see that intuitionistic (implicational) logic is obtained by considering success condition (s1) and no other constraint. More interestingly, to make a connection with the previous chapter, we can see that **S4** strict implication is given by considering success condition (s1) and restrctions (r0) and (r2) on reduction. We leave to the reader to prove that the above formulation coincide with the database-as-list formulation of **S4** we have seen in the previous chapter. We can consider **S4**, as a substructural logic, which is obtained by imposing a restriction on the weakening and the permutation rules. On the other hand, **S4** is strongly related to relevance logic **E**, which is regarded as a logic of relevance and necessity [**?**].

We introduce some notational convention which we will use later on. We use the notation

$$Q \Rightarrow_z^{Red} Q_1, \ldots, Q_n,$$

to denote that query Q may generate queries $Q_1, \ldots, Q_n$, by reduction wrt some

$$z : D_1 \rightarrow \ldots \rightarrow D_n \rightarrow q \in \Gamma.$$

Similarly, let $Q = \Gamma, \gamma \ \vdash^? \ x : A \rightarrow B$, if from Q we step to Q' by the implication rule, where:

$$Q' = \ \Delta \cup \{y : A\}\delta \cup \{y\} \ \vdash^? \ y : B,$$

for some $y > max(Lab(\Delta))$, (whence $y \notin Lab(\Delta)$), we write

$$Q \Rightarrow_y^{Imp} Q'.$$

Given a resource set γ and a label u, let us define:

$$pred(\gamma, u) = \begin{cases} max\{z \in \gamma \mid z < u\} \\ \uparrow \text{ if } \neg\exists z \in \gamma \ z < u. \end{cases}$$

$$succ(\gamma, u) = \begin{cases} min\{z \in \gamma \mid z > u\} \\ \uparrow \text{ if } \neg\exists z \in \gamma \ z > u. \end{cases}$$

By convention, we assume that every inequality in which $pred(\gamma, u)$ ($succ(\gamma, u)$) is one of the two terms trivially holds if $pred(\gamma, u) = \uparrow$ ($succ(\gamma, u) = \uparrow$); similarly, we assume that an inequality such as $max\{pred(\gamma, u), x\} < y$ reduce to $x < y$, whenever $pred(\gamma, u)$ is undefined.

5.3 Properties

In this section we list some properties of the computation procedure which allow some simplification. The proofs are simple by induction on the height of successful computation and will be mostly omitted.

Proposition 5.3.1 *If* $\Gamma, \gamma \ \vdash^? \ x : B$ *succeeds,* $u : A \in \Gamma$*, but* $u \notin \gamma$*, then*

$$\Gamma - \{u : A\}, \gamma \ \vdash^? \ x : B \textit{ succeeds.}$$

By this proposition is apparent that γ specifies usage. Resources which are not included in γ are not relevant for the computation and the formulas they label can be omitted.

Proposition 5.3.2 *If* $\Gamma, \gamma \vdash^? x : B$ *succeeds, then* $\gamma \subseteq Lab(\Gamma)$

By the previous proposition, when we are concerned with successful queries, we can assume that $\gamma \subseteq Lab(\Gamma)$, and hence $max(\gamma) \leq max(Lab(\gamma))$.

The following proposition generalizes the success rule to the identity property.

Proposition 5.3.3 *If* $x : A \in \Gamma$, *and* $Succ^S(\gamma, x)$, *then* $\Gamma, \gamma \vdash^? x : A$ *succeeds.*

Proof. By induction on the complexity of A. if $cp(A) = 0$, that is A is an atom, the one above is just the success rule. Let

$$A = B_1 \rightarrow \ldots \rightarrow B_n \rightarrow q$$

A derivation of A, will start with:

$$\Gamma, \gamma \vdash^? x : A$$

$$\downarrow$$

$$(Q)\ \Gamma \cup \{z_1 : B_1, \ldots, z_n : B_n\}, \gamma \cup \{z_1, \ldots, z_n\} \vdash^? z_n : q$$

where $max(Lab(\Gamma)) < z_1 < \ldots z_n$. Let $\Gamma' = \Gamma \cup \{z_1 : B_1, \ldots, z_n : B_n\}$ and $\gamma' = \gamma \cup \{z_1, \ldots, z_n\}$. We simply define

$\gamma_0 = \{x\}$ and $z_0 = x$,
$\gamma_1 = (\gamma - \{x\}) \cup \{z_1\}$
$\gamma_i = \{z_i\}$, for $i > 1$.

We can easily see that $Succ^S(\gamma_i, z_i)$, hence, by induction hypothesis we have that

all $(Q_i)\ \Gamma', \gamma_i \vdash^? z_i : B_i$ succeed.

If we can apply reduction wrt. $x : A$ to (Q), step to queries (Q_i), and we are done. But, by definition $z_0 = x$, $\gamma_0 = \{x\}$, and $\bigcup_{i=0}^n \gamma_i = \gamma'$. Moreover, we leave to the reader to check that

$$Red^S(\gamma_0, \ldots, \gamma_k, z_0, \ldots, z_n; z_n)$$

holds, so that we can perform the above reduction step. This complete the proof. □

Proposition 5.3.4 *If* $\Gamma, \gamma \vdash^? x : B$ *succeeds,* $\Gamma \subseteq \Delta$, *then*

$\Delta, \gamma \vdash^? x : B$ *succeeds.*

In case of **BCK** *if* $\gamma \subseteq \delta$ *also*

$\Delta, \delta \vdash^? x : B$ *succeeds.*

Moreover, the height of a successful derivation does not increase.

Proposition 5.3.5

- *For* **R**, **L**, **BCK**, *if* $\Gamma, \gamma \vdash^? x : B$ *succeeds, then*

  for every $y \in \gamma$, $\Gamma, \gamma \vdash^? y : B$ *succeeds.*

- *For* **R**, **L**, **BCK**, **E**, **E-W**, *if* $\Gamma, \emptyset \vdash^? x : B$ *succeeds, then*

  for every y, $\Gamma, \emptyset \vdash^? y : B$ *succeeds.*

- *For* **E**, **E-W**, *if* $\Gamma, \gamma \vdash^? x : B$ *succeeds and* $x \notin \gamma$, *then,*

  $$\text{for every } y \geq max(\gamma),\ \Gamma, \gamma \vdash^? y : B \text{ succeeds.}$$

Moreover, the height of a successful derivation does not increase.

Because of the previous proposition, in case of **R**, **L**, and **BCK** we can omit the label x in front of the goal. However, we do not make use of this and other possible simplification at this stage since we want to carry on an uniform development.

For some technical reasons, connected with the proof of cut-admissibility theorem in the next section, we restrict our consideration to a certain type of queries, namely queries which can be generated asking an initial goal from the empty database.

Definition 5.3.6 We say that a query

$$\Gamma, \gamma \vdash^? x : A$$

is S-*regular*, where S in one of the logic under consideration if the following holds:

- for **R**, **L**, **BCK**, if $\gamma \neq \emptyset$, then $x \in \gamma$;
- for **E**, **E-W**, $max(\gamma) \leq x$,
- for **T**,**T-W**, **CL**, $max(\gamma) = x$ (thus if $\gamma = \emptyset$, then $x = \mathbf{v_0}$).

An S-*regular* derivation is a derivation in which every query is S-regular. ■

In particular, notice that a query from the empty database is S-regular, for every system S. Without loss of generality, we can to restrict our attention to regular queries and regular derivations.

Proposition 5.3.7 *Let Q be an S-regular query, if Q succeeds, then it succeeds by an S-regular derivation.*

We conclude this section by showing that successful queries are closed under *formula substitution.* Here by formula substitution we mean the simultaneous substitution of all occurrences of an atom q whithin a formula $A[q]$, by an arbitrary formula B. The result of the substitution is denoted by $A[q/B]$. Similarly, we denote by $\Gamma[q/B]$, the substitution of q by B in every formula of a database Γ. The property of closure under substitution is not to be taken for granted, as the deduction rules pay attention to the form of the formulas, that is whether they are atomic or not.

Theorem 5.3.8 *If $Q = \Gamma, \gamma \vdash^? x : A$ succeeds, then also $Q' = \Gamma[q/B], \gamma \vdash^? x : A[q/B]$ succeeds.*

Proof. By induction on the height h of a successful derivation of the query Q. If $h = 0$, then the query immediately succeeds. If $A \neq q$, the claim is trivial, if $A = q$, then $x : B \in \Gamma[q/B]$, and the claim follows by proposition 5.3.3.

Let $h > 0$ if $A = C \to D$, then we apply the implication rule and we step to

$$\Gamma \cup \{z : C\}, \gamma \cup \{z\} \vdash^? z : D,$$

for a suitable z. By induction hypothesis, we have that

$$(\Gamma \cup \{z : C\})[q/B], \gamma \cup \{z\} \vdash^? z : D[q/B] \text{ succeeds.}$$

and we can conclude by applying the implication rule to Q' that is $\Gamma[q/B], \gamma \vdash^? x : A[q/B]$ as $A[q/B]$ is $C[q/B] \to D[q/B]$ and $(\Gamma \cup \{z : C\})[q/B]$ is $\Gamma[q/B] \cup \{C[q/B]\}$.
Let A be an atom r and the first rule applied be reduction with respect to $z : C \in \Gamma$, where $C = D_1 \to \ldots \to D_n \to r$, so that we step to

$$Q_i = \Gamma, \gamma_i \vdash^? x_i : D_i$$

for some suitable γ_i and x_i such that $\gamma = \bigcup_i \gamma_i$ and $Red^S(\gamma_0, \ldots, \gamma_n, x_0, \ldots, x_n; x)$ holds; if $r \neq q$, we simply apply the induction hypothesis to Q_i and we get that

$\Gamma[q/B], \gamma_i \vdash^? x_i : D_i[q/B]$ succeed.

We can apply reduction to Q' wrt. $z : C[q/B]$ and succeed. If $r = q$, letting $B = B_1 \to \ldots \to B_m \to p$, we have that $C[q/B]$ is

$$D_1[q/B] \to \ldots \to D_n[q/B] \to B_1 \to \ldots \to B_m \to p.$$

Then from Q', we step by the implication rule to

$$Q'' = \Gamma[q/B] \cup \{z_1 : B_1, \ldots, z_m : B_m\}, \gamma \cup \{z_1, \ldots, z_m\} \vdash^? z_m : p$$

where $max(Lab(\Gamma[q/B])) < z_1 < \ldots < \ldots z_m$. Let $\Gamma' = \Gamma[q/B] \cup \{z_1 : B_1, \ldots, z_n : B_m\}$ and $\gamma' = \gamma \cup \{z_1, \ldots, z_m\}$. Let $\gamma_{n+j} = \{z_j\}$, and $x_{n+j} = z_j$ for $j = 1, \ldots, m$. It is easy to see that

$\gamma' = \bigcup_{i=0}^{n+m} \gamma_i$, and also $Red^S(\gamma_0, \ldots, \gamma_{n+m}, x_0, \ldots, x_{n+m}; z_m)$ holds,

by the hypothesis that $Red^S(\gamma_0, \ldots, \gamma_n, x_0, \ldots, x_n; x)$ holds. Thus, from Q'' we can step to

$Q'_i = \Gamma', \gamma_i \vdash^? x_i : D_i[q/B]$, for $i = 1, \ldots, n$ and
$Q'_{n+j} = \Gamma', \gamma_{n+j} \vdash^? x_{n+j} : B_j$, for $j = 1, \ldots, m$

Since the queries Q_i succeed, by induction hypothesis and proposition 5.3.4 also the queries Q'_i succeed. On the other hand $Succ^S(\gamma_{n+j}, x_{n+j})$ hold, and $x_{n+j} : B_j \in \Gamma'$, thus by proposition 5.3.3, we have that queries Q_{n+j} succeed as well, thus query Q'' succeed and so does query Q'. $\square$

5.4 Admissibility of Cut

In this section we prove the admissibility of cut for all the logics under consideration. We first have to deal with the notion of substitution of formulas and labels by databases. In this case, the proof of the admissibility of cut is more complex than in the cases we have seen in the previous chapters. The reason is that the deduction rules are more constrained than in the previous cases. In particular we have to take into account the ordering of formulas in the databases involved in a cut inference step.

Since we treat in this section the pure implicational case, we need the notion of compatibility with respect to substitution, that we have already met in the previous chapters.

Definition 5.4.1 Given two databases Γ, with $u : A \in \Gamma$ and Δ, we say that Γ and Δ are *compatible for substitution* if $z \in Lab(\Gamma) \cap Lab(\Delta)$, then $z : C \in \Gamma$ iff $z : C \in \Delta$, and whenever they are compatible for substitution, we let:

$$\Gamma[u : A/\Delta] = (\Gamma - \{u : A\}) \cup \Delta.$$

Given two sets of resources γ and δ and a label u we define:

$$\gamma[u/\delta] = \begin{cases} (\gamma - \{u\}) \cup \delta \text{ if } u \in \gamma \\ \gamma \text{ otherwise.} \end{cases}$$

■

By cut we mean the following inference rule: If

$$\Gamma[u : A], \gamma \vdash^? x : B \quad \text{and} \quad \Delta, \delta \vdash^? y : A$$

succeed, then also

$$\Gamma[u : A/\Delta], \gamma[u/\delta] \vdash^? x[u/y] : B$$

succeeds.

However, unless we put some constraint on γ and δ the above rule does not hold. To give a simple example, let us consider the case of **L** (linear implication), we have:

(1) $\{x_2 : p \to q, \mathbf{x_3 : p}\}, \{x_2, x_3\} \vdash^? x_3 : q$ and
(2) $\{x_1 : (p \to q) \to p, x_2 : p \to q\}, \{x_1, x_2\} \vdash^? \mathbf{x_2 : p}$ both succeed, but
(3) $\{x_1 : (p \to q) \to p, x_2 : p \to q\}, \{x_1, x_2\} \vdash^? x_2 : q$ fails.

Here we want to replace $\mathbf{x_3 : p}$ occurring in (1) by the database in (2). Since the database shares resource x_2, $x_2 : p \to q$ must be used *twice* in a deduction of (3), and this is not allowed by the condition for **L**-computation. In this case, the restriction we must put is that the two resource sets (in the two premises of the cut) γ and δ be *disjointed.* For other logics, for instance, where the order matters, we must ensure that the substitution of δ in γ respects the ordering of the labels in γ. However, we can always fulfill these compatibility constraints by renaming the labels in the proper way. In the previous example, for instance, we can rename the labels in the first query, and get

(1') $\{y_2 : p \to q, \mathbf{y_3 : p}\}, \{y_2, y_3\} \vdash^? y_3 : q$

by cut we now obtain:

(3') $\{x_1 : (p \to q) \to p, x_2 : p \to q, y_2 : p \to q\} \{x_1, x_2, y_2\} \vdash^? x_2 : q$

which succeeds.

We hence have two alternatives, either we accept the above formulation of the cut rule, but we restrict its applicability to queries which satisfy specific coditions for each system, or we find a more general notion of cut which hold in every case. As we will see, we can define the more general notion of cut as well by incorporating the needed re-labelling. However, we will prove the admissibility of the cut in the restricted form, since in some sense is more informative. For instance, from the premises (1) and (2) above , we can legitamately infer the conclusion (3) in **R**, **T**, **E**, and not only the weaker (3').

We introduce some notation to simplify the following development. Given two queries Q, Q', with $Q = \Gamma[u : A], \gamma \vdash^? x : A$ and $Q' = \Delta, \delta \vdash^? y : A$, no matter whether they are compatible or not, the result of cutting Q by Q' on $u : A$ is uniquely determined, and we denote it by the query

$Q'' = CUT(Q, Q', u)$.

Definition 5.4.2 [Compatibility for Cut] Given two queries

$Q = \Gamma[u : A], \gamma \vdash^? x : A$ and $Q' = \Delta, \delta \vdash^? y : A$, we say that Q and Q' are *compatible for cut* on position/resource u in system S, denoting this fact by $COMP^S(Q, Q', u)$ [4], if the following holds:

- Γ and Δ are compatible for substitution;
- the following combinations of the conditions (1)-(3) below hold.
  1. $\gamma \cap \delta = \emptyset$;
  2. $pred(\gamma, u) < min(\delta)$ and $max(\delta) < succ(\gamma, u)$;
  3. $pred(\gamma, u) \leq y$, and if $u < x$, then $y \leq min\{succ(\gamma, u), x\}$.

  - for **R**: nothing;
  - for **L**, **BCK**: (1);
  - for **CL**: (2);
  - for **T**, **E**: (3);
  - for **T-W**, **E-W**: (1) and (3).

■

Some explanation of these conditions helps: condition (1) says that the resources of the two premises are disjointed and this condition must be assumed for systems without contraction. Condition (2) is stronger and requires that δ seen as an ordered sequence of labels must fit in between the predecessor and the successor of the label u that we replace by cut; in other words, u can be replaced by the entire 'segment' δ without mixing γ and δ. Condition (3) is a weakening of (2) allowing merging of γ and δ up to the successor of the cut formula label u. The condition say exactly this, whenever $succ(\gamma, u)$ exists, whence $succ(\gamma, u) \leq x$), and $y = max(\delta)$. It is stated in more general terms, since $succ(\gamma, u)$ may not exist, and in the cases of **E** and **E-W** it may be $y \geq max(\delta)$. For instance, let $x_1 < x_2 < \ldots < x_9$, two queries with $\gamma = \{x_2, x_4, x_6, x_8, x_9\}$, $u = x_6$, and $x = x_9$, $\delta = \{x_1, x_3, x_6, x_7\}$, $y = x_7$, satisfy condition (3), whereas if $\delta = \{x_1, x_3, x_6, x_9\}$, they do not satisfy it.

It should also be clear that, if $y = max(\delta)$ then condition (2) implies both condition (1) and condition (3).

The compatibility conditions preserve the regularity of the queries.

Proposition 5.4.3 *If Q and Q' are S-regular and $COMP^S(Q, Q', u)$ holds, then the query $Q'' = CUT(Q, Q', u)$ is S-regular.*

Proof. We proceed by cases, let first assume $\gamma[u/\delta] \neq \emptyset$.

- Case of **R**, **L**, **BCK**, we must show that $x[u/y] \in \gamma[u/\delta]$. By regularity of Q and Q', we have $x \in \gamma$ and $y \in \delta$. If $x = u$, then $x[u/y] = y \in \delta \subseteq \gamma[u/\delta]$.

  If $x \neq u$, then $x = x[u/y] \in \gamma - \{u\} \subseteq \gamma[u/\delta]$.
- Case of **E**, **E-W**, we must show that $max(\gamma[u/\delta]) \leq x[u/y]$. We have the following cases:
  1. Let $\gamma[u/\delta] = \gamma$ and $x[u/y] = x$, then it follows by the regularity of Q.

[4]As can be seen from the conditions below, that compatibility is a relation on (sets of) labels which occur in a query, the formulas do not play any role, we will use later on the notation $Comp^S(\Gamma, \gamma, x, u; \Delta, \delta, y)$.

2. Let $\gamma[u/\delta] = (\gamma - \{u\}) \cup \delta$ and $x[u/y] = x$. We have that $max(\gamma[u/\delta]) = max(\gamma - \{u\})$ or $max(\gamma[u/\delta]) = max(\delta)$. In the former case we conclude by regularity of Q. In the latter case since $u \in \gamma$ and $x \neq u$, by regularity of Q it must be $u < x$, so that by $COMP^S(Q, Q', u)$, and by regularity of Q' we have

   $\max(\delta) \leq y \leq min\{succ(\gamma, u), x\} \leq x.$

3. $\gamma[u/\delta] = \gamma$ and $x[u/y] = y$, that is $x = u$. We have $u \notin \gamma$. It must be $pred(\gamma, u) \downarrow$, for otherwise, by regularity of Q we would have $max(\gamma) < u$, and hence $\gamma = \emptyset$ against the hypothesis. Since $pred(\gamma, u) \downarrow$, we have $max(\gamma) = pred(\gamma, u) \leq y$, by $COMP^S(Q, Q', u)$.

4. Let $\gamma[u/\delta] = (\gamma - \{u\}) \cup \delta$ and $x[u/y] = y$, that is $x = u$. Then $max(\gamma[u/\delta]) = max(\gamma - \{u\})$ or $max(\gamma[u/\delta]) = max(\delta)$. In the former case, we have $max(\gamma - \{u\}) = pred(\gamma, u) \leq y$, by $COMP^S(Q, Q', u)$. In the latter case we conclude by the regularity of Q'.

- For all the other systems we checks in a similar way that $max(\gamma[u/\delta]) = x[u/y]$. Details are left to the reader.

- For **T**, **T-W**, **CL**, we check that if $\gamma[u/\delta] = \emptyset$, then $x[u/y] = \mathbf{v_0}$. If $\gamma[u/\delta] = \emptyset$, then either $\gamma = \emptyset$ or $\gamma = \{u\}$ and $\delta = \emptyset$. In the former case, we have $x = \mathbf{v_0}$, so that it must be $\mathbf{v_0} \neq u$, since $\mathbf{v_0} \notin Lab(\Gamma)$, and the result follows. In the latter case, it must be $x = u$, so that $x[u/y] = y = \mathbf{v_0}$, as $\delta = \emptyset$.

□

The proof of the admissibility of cut relies on the lemmas below, which state properties and relations of $COMP^S$, Red^S, $Succ^S$.

We give an intuitive explanation of the lemmas. Lemma 5.4.4 deals with the case when the cut formula is an atom and it is used for immediate success. Lemmas 5.4.5 and 5.4.6 ensures the permutability of cut by the implication rule and the reduction rule, respectively. Lemma 5.4.8 deals with the case when the cut formula is used in a reduction step, that is it is the principal formula. To explain the role of lemmas 5.4.6 and 5.4.8, we illustrate the cut elimination process in some detail in this last case; we do not mention the labels, but we focus on the inference steps. Let us have

(1) $\Gamma, D \to q \vdash^? q$ and (2) $\Delta \vdash^? D \to q$

From (1) we step to

(3) $\Gamma, D \to q \vdash^? D$.

By cutting (2) and (3) we obtain

(4) $\Gamma, \Delta \vdash^? D$ succeeds.

By (2), we get:

(5) $\Delta, D \vdash^? q$ succeeds.

Now we cut (5) and (4) on D and obtain:

(6) $\Gamma, \Delta \vdash^? q$ succeeds.

Since, we have labels and constraints, we cannot freely perform cut, we must ensure that compatibility constraints are satisfied. The meaning of the lemmas is that: by 5.4.6 if (1) and (2) are compatible for cut, then so are (3) and (2); by 5.4.8 if (1) and (2) are compatible for cut, then so are (5) and (4). The structure of cut elimination process is fixed and it is exactly the same as in the previous chapters. Much of the effort is due to ensure the right propagation of compatibility constraints. To simplify, we assume that the query are S-regular. Success always mean success with respect to a system S.

Lemma 5.4.4 *If $\Delta, \delta \vdash^? y : q$ succeeds and $Succ^S(\gamma, u)$ holds, then $\Delta, \gamma[u/\delta] \vdash^? y : q$ succeeds.*

Proof. Easy and left to the reader. $\square$

Lemma 5.4.5 *If $Q \Rightarrow_z^{Imp} Q_1$, and $COMP^S(Q, Q', u)$, then there exist a label z', such that $Q \Rightarrow_{z'}^{Imp} Q_1[z/z']$, $COMP(Q_1[z/z'], Q', u)$, and*

$$CUT(Q, Q', u) \Rightarrow_{z'}^{Imp} CUT(Q_1[z/z'], Q', u).$$

Proof. Let $Q = \Gamma, \gamma \vdash^? x : C \to D$, choose z' such that $max\{max(\gamma), max(\delta), x, u, y\} < z'$, and let $Q_1 = \Gamma \cup \{z' : C\}\delta \cup \{z'\} \vdash^? z' : D$, we have that $CUT(Q_1, Q', u)$ is the query

$$(\Gamma \cup \{z' : C\})[u : A/\Delta], (\delta \cup \{z'\})[u/\delta] \vdash^? z'[u/y] : D.$$

By the choice of z', $z' \neq u$ and $z' \notin \gamma \cup \delta$, we have that the above query is the same as:

$$\Gamma[u : A/\Delta] \cup \{z' : C\}), \delta[u/\delta] \cup \{z'\} \vdash^? z' : D.$$

Thus, it is clear that

$$CUT(Q, Q', u) \Rightarrow_{z'}^{Imp} CUT(Q_1, Q', u).$$

We have still to check that $COMP^S(Q_1, Q', u)$. As far as condition (1) is concerned, if $\gamma \cap \delta = \emptyset$, also $(\gamma \cup \{z'\}) \cap \delta = \emptyset$ holds, since $z \notin \delta$.

For condition (2), the only non-trivial check is that $y < succ(\gamma \cup \{z'\}, u)$ when $succ(\gamma \cup \{z'\}, u) = z'$, but this is ensured by the choice of $z' > y$.

Smilarly, for condition (3), the only non-trivial check is $y \leq min\{succ(\gamma \cup \{z'\}), z'$, which holds again as $z' > y$. $\square$

Lemma 5.4.6 *Suppose $Q \Rightarrow_z^{Red} Q_1, \ldots, Q_n$, where*

$$Q = \Gamma[u : A], \gamma \vdash^? x : q$$
$$Q_i = \Gamma[u : A], \gamma_i \vdash^? x_i : D_i, \text{ and } Q' = \Delta, \delta \vdash^? y : A.$$

If $COMP^S(Q, Q', u)$ then

- *(a) $COMP^S(Q_i, Q', u)$;*
- *(b) $z \neq u$ implies $CUT(Q, Q', u) \Rightarrow_z^{Red} CUT(Q_1, Q', u), \ldots, CUT(Q_n, Q', u)$*

Proof. We first show that we can assume that $x_i \in \gamma$ or $x_i = x$. For all systems, but **E** and **E-W**, this is obvious: by regularity, $x_i \in \gamma_i \subseteq \gamma$, unless $\gamma_i = \emptyset$ (and this is allowed in some systems). But if $\gamma_i = \emptyset$ is allowed (and it is so in systems **R**, **BCK**), whatever is the choice of x_i, by proposition 5.3.5 the query succeeds with the same height; thus we may chose a proper $x_i \in \gamma$ or let $x_i = x$.

In case of **E** and **E-W**, if $x_i \notin \gamma_i \subseteq \gamma$, by property 5.3.5, we have that for every $w \geq max(\gamma_i)$, $\Gamma, \gamma_i \vdash^? w : D_i$ succeeds, so that again we can chose a proper $x_i \in \gamma$ or take $x_i = x$.

We turn to prove (a), for each group of systems.

- (Condition c1) We must check that $\gamma_i \cap \delta = \emptyset$. But this trivially follows from the fact that $\gamma \cap \delta = \emptyset$ and $\gamma_i \subset \gamma$.
- (Condition c2) We must check that $pred(\gamma_i, u) < min(\delta)$ and $max(\delta) < succ(\gamma_i, u)$. Notice that there is only one i_0 such that $u \in \gamma_{i_0}$ and by (r4), if $pred(\gamma_{i_0}, u) \downarrow$, then $pred(\gamma_{i_0}, u) = pred(\gamma, u)$, and the same holds for $succ(\gamma_{i_0}, u)$.
- (Condition c3) We must check that

  (*) $pred(\gamma_i, u) \leq y$ and if $u < x_i$, then $y \leq min\{succ(\gamma_i, u), x_i\}$.

  Notice that since $\gamma_i \subseteq \gamma$, we have

  (1) if $pred(\gamma_i, u) \downarrow$, then $pred(\gamma, u) \downarrow$ and $pred(\gamma_i, u) \leq pred(\gamma, u)$;
  (2) if $succ(\gamma_i, u) \downarrow$, then $succ(\gamma, u) \downarrow$ and $succ(\gamma_i, u) \geq succ(\gamma, u)$.

  Then, (*) easily follow from (1), (2), $COMP(Q, Q', u)$, and the regularity of Q_i, which implies by (2), $succ(\gamma_i, u) \leq x_i$.

We now turn to (b) We must show that if $u \neq z$, then the following hold:

1. $\gamma_0[u/\Delta] = \{z\}$ and $x_0[u/y] = x_0$,
2. $\gamma[u/\delta] = \bigcup_{i=0}^{n} \gamma_i[u/\delta]$,
3. $Red^S(\gamma_0[u/\delta], \ldots, \gamma_n[u/\delta], x_0[u/y], \ldots, x_n[u/y]; x[u/y0)$

Condition 1. is obvious by $u \neq z$. For 2., we know that $\gamma_0 = \{z\}$. Let $u \in \gamma$ (otherwise the claim is trivial), then for some i it must be $u \in \gamma_i$. Let $J = \{j : 1, \ldots, n \mid u \in \gamma_j\}$, and let $J^c = \{1, \ldots, n\} - J$. Then we have:

$$\gamma[u/\delta] = (\bigcup_{i=0} \gamma_i)[u/\delta] = (\bigcup_{i \in J} \gamma_i)[u/\delta] \cup (\bigcup_{j \in J^c} \gamma_j[u/\delta]),$$

then the result easily follows from the fact that

for $j \in J^c$,$\gamma_j[u/\delta] = \gamma_j$ and $(\bigcup_{i \in J} \gamma_i)[u/\delta] = \bigcup_{i \in J}(\gamma_i[u/\delta])$.

For 3., we have to check (r0)–(r4). We give the details of the cases (r2) and (r4), the proof of the others are similar or straightforward.

- (r2) $max(\gamma_i[u/\delta]) \leq x_i[u/y]$ follows by proposition 5.4.3. We must prove $x_{i-1}[u/y] \leq x_i[u/y]$, for $i = 1, \ldots, n$.
  - Let $x_{i-1}[u/y] = x_{i-1}$, and $x_i[u/y] = y$, that is $x_i = u$; then $x_{i-1} < u$, we know that $x_{i-1} \in \gamma$, (it cannot be $x_{i-1} = x$, for otherwise, we would have $x < u$). Hence $pred(\gamma, u) \downarrow$, so that, by $COMP^S(Q, Q', u)$, we have:

    $x_{i-1} \leq pred(\gamma, u) \leq y$.
  - Let $x_{i-1}[u/y] = y$ and $x_i[u/y] = x_i$, that is $x_{i-1} = u$, By part (a), we have $COMP^(Q_i, Q', u)$, so that, being $u < x_i$, we have

    $y \leq min\{succ(\gamma_i), x_i\} \leq x_i$.
- (r4) Only one γ_i and x_i, say for $i = i'$ are affected by substitution by δ and y (i.e. for only one i, $u \in \gamma_i$), and since $z \neq u$, $i' > 0$. In light of 5.4.3, we must prove only that:

$max(\gamma_{i'-1}) < min(\gamma_{i'}[u/\delta])$ and
$max(\gamma_{i'}[u/\delta]) < min(\gamma_{i'+1})$

The first claim is not trivial in the case $u = min(\gamma_{i'})$, but in such a case, by $COMP(Q, Q', u)$ we have:

$$max(\gamma_{i'-1}) = pred(\gamma, u) < min(\delta) = min(\gamma_{i'}[u/\delta]).$$

Similarly, the second claim is not trivial in case $u = max(\gamma_{i'})$, but in such a case, by $COMP(Q, Q', u)$ we have:

$$max(\gamma_{i'}[u/\delta]) = max(\delta) < succ(\gamma, u) = min(\gamma_{i'+1}).$$

□

Remark 5.4.7 Notice that the proof of $x_{i-1}[u/y] \leq x_i[u/y]$ for (r2) and (r3) does not depend on the assumption of $z \neq u$, that is the claim hold even if $u = z$. This fact will be useful in the proof of lemma 5.4.8.

Lemma 5.4.8 *Let the following hold:*

(i) $Q = \Gamma[u : D_1 \to \ldots \to D_k \to q], \gamma \vdash^{?} x : q$;
(ii) $Q' = \Delta, \delta \vdash^{?} y : D_1 \to \ldots \to D_k \to q$;
(iii) $Q \Rightarrow_u^{Red} Q_1, \ldots, Q_n$,
(iv) $COMP^S(Q, Q', u)$,
(v) $Q'_i = CUT(Q_i, Q', u)$ *for* $i = 1, \ldots, n$,
(vi) $Q''_0 = \Delta \cup \{z_1 : D_1, \ldots, z_n : D_n\}\delta \cup \{z_1, \ldots, z_n\} \vdash^{?} z_n : q$, *where*

$$max\{max(\gamma), max(\delta), x, y, u\} < z_1 < z_2 < \ldots < z_k,$$

(vii) $Q''_i = CUT(Q''_{i-1}, Q'_i, z_i)$ *for* $i = 1, \ldots, n$,

then, $COMP^S(Q''_{i-1}.Q'_i, z_i)$ *holds for* $i = 1, \ldots, n$.

Proof. Let $A = D_1 \to \ldots \to D_k \to q$. We define:

$$\delta_i = \delta \cup \{z_{i+1}, \ldots, z_n\} \cup \bigcup_{j=1}^{i} \gamma_j[u/\delta]$$

(so that $\delta_0 = \delta \cup \{z_1, \ldots, z_n\}$). Thus, we have:

$Q'_i = \Gamma[u : A/\Delta], \gamma_i[u/\delta] \vdash^{?} x_i[u/y] : D_i$, and
$Q''_i = (\Gamma - \{u : A\}) \cup \Delta \cup \{z_{i+1} : D_{i+1}, \ldots, z_n : D_n\}, \delta_i \vdash^{?} z_n : q$,

(for the sake of uniformity, by proposition 5.3.4, we can replace Δ by $(\Gamma - \{u : A\}) \cup \Delta$ in Q''_0), We must prove $COMP^S(Q''_{i-1}.Q'_i, z_i)$, for $i = 1, \ldots, n$, checking each condition involved in $COMP^S$.

- (condition c1) for systems with (r1). We have that $u \notin \gamma_i$, hence for all i, $\gamma_i[u/\delta] = \gamma_i$; since $\delta \cap \gamma_i = \emptyset$, by the choice of z_i, we also get that for $i = 0, \ldots, n-1$

$$(\delta \cup \{z_{i+1}, \ldots, z_n\} \cup \bigcup_{j=1}^{i} \gamma_j[u/\delta]) \cap \gamma_{i+1} = \emptyset,$$

- (condition c2) Observe that $u = \min(\gamma)$ and $u \notin \gamma_i$, whence for all i, $\gamma_i[u/\delta] = \gamma_i$. We must check that

$$pred(\delta_i, z_{i+1}) < min(\gamma_{i+1}) \text{ and } max(\gamma_{i+1}) < succ(\delta_i, z_{i+1}).$$

The second inequality holds by the choice of z_i. For the first inequality, in case of $i = 0$, we have that

$$pred(\delta_0, z_1) = max(\delta) < succ(\gamma, u) = min(\gamma_1),$$

by $COMP^S(Q, Q', u)$. In case of $i > 0$, we have

$$pred(\delta_i, z_{i+1}) = max(\delta \cup \bigcup_{j=1}^{i} \gamma_j) = max(\gamma_i) < min(\gamma_{i+1}).$$

- (condition c3) We must check that

  $pred(\delta_i, z_{i+1}) \leq x_{i+1}[u/y]$ and
  if $z_{i+1} < z_n$, then $x_{i+1}[u/y] \leq min\{succ(\delta_i, z_{i+1})), z_n\}$.

  The second inequality holds by the choice of z_i's. We prove the first one in case of system **E**; for the other systems the proof is almost the same. Let $i = 0$, then we have

  $$pred(\delta_0, z_1) = max(\delta) \leq y.$$

  We know that $u \leq x_1$, thus if $u = x_1$, we are done. Let $u < x_1$, then $x_1[u/y] = x_1$. By $COMP^S(Q, Q', u)$ and by proposition 5.4.6, we have $COMP^S(Q_1, Q', u)$, whence

  $$max(\delta) \leq y \leq min\{succ(\gamma_1, u), x_1\} \leq x_1.$$

  Let $i > 0$, then we have:

  $$pred(\delta_i, z_{i+1}) = max(\delta \cup \bigcup_{j=1}^{i} \gamma_j)[u/\delta]).$$

  By remark 5.4.7, we know that $x_1[u/y] \leq \ldots \leq x_n[u/y]$. Then we have that either:

  $$max(\delta \cup \bigcup_{j=1}^{i} \gamma_j)[u/\delta]) = max(\delta),$$

  so that $max(\delta) \leq x_1[u/y] \leq x_{i+1}[u/y]$, or

  $$max(\delta \cup \bigcup_{j=1}^{i} \gamma_j)[u/\delta]) = max(\gamma_j[u/\delta]), \text{ for some } j \leq i.$$

  But by proposition 5.4.3 $max(\gamma_j[u/\delta]) \leq x_j[u/y] \leq x_i[u/y]$.

□

Theorem 5.4.9 (Admissibility of Cut) *Let* $Q = \Gamma[u : A], \gamma \vdash^? x : B$ *and* $Q' = \Delta, \delta \vdash^? y : A$ *be regular queries. If* Q *and* Q' *succeed and* $COMP^S(Q, Q', u)$*, then also*

$$Q^* = CUT(Q, Q', u) = \Gamma[u : A/\Delta], \gamma[u/\delta] \vdash^? x[u/y] : B \textit{ succeeds.}$$

Proof. Let Q and Q' be as in the statement of the theorem. As usual, we proceed by double induction on pairs (c, h), where c is the complexity of the cut formula A and h is a successful derivation of Q. We give the details for the case of immediate success ($c = h = 0$), and for the case of reduction with respect to $u : A$ ($c, h > 0$). The other cases follow by induction hypothesis using lemma 5.4.5 if the first step is by the implication rule), and using lemma 5.4.6 if the first step is by the reduction rule.

Let $c = 0$ and $h = 0$, then Q immediately succeeds, that is B is an atom q, $x : q \in \Gamma$ and $Succ^S(\gamma, x)$ holds. If $x \neq u$, then $x : q$ is also in $\Gamma[u : A/\Delta]$, moreover $x[u/y] = x$ and $\gamma[u/\delta] = \gamma$. By propositions 5.3.1 and 5.3.4 we have that $Q^* = \Gamma - \{u : A\} \cup \Delta, \gamma[u/\delta] \ \vdash^? \ x[u/y] : B$ succeeds. If $x = u$, then $x[u/y] = y$, since Q' succeeds by lemma 5.4.4, we get that Q^*, succeeds.

Let $c, h > 0$ and Q succeeds by reduction with respect to $u : A$. Let $A = D_1 \to \ldots \to D_n \to q$. We have $Q \Rightarrow_u^{Red} Q_1, \ldots, Q_n$,which succeed with smaller height; by lemma 5.4.6 we have $COMP^S(Q_i, Q', u)$, hence by induction hypothesis we get that $Q'_i = CUT(Q_i, Q', u)$ succeeds. Now let,

$Q_0^* = \Delta \cup \{z_1 : D_1, \ldots, z_n : D_n\}, \delta \cup \{z_1, \ldots, z_n\} \ \vdash^? \ z_n : q$, where
$max\{max(\gamma), max(\delta), x, y, u\} < z_1 < z_2 < \ldots z_k$;

by lemma 5.4.8 we have $COMP(Q_0^*, Q'_1, z_1)$, notice that $cp(D_1) < cp(A) = c$, thus we may apply the induction hypothesis and obtain that $Q_1^* = CUT(Q_0^*, Q'_1, z_1)$ that is

$\Gamma - \{u : A\} \cup \Delta \cup \{z_2 : D_2, \ldots, z_n : D_n\}, \gamma_1[u/\delta] \cup \delta \cup \{z_2, \ldots, z_n\} \ \vdash^? \ z_n : q$

succeeds. By lemma 5.4.8 also $COMP(Q_1^*, Q'_2, z_2)$, and since $cp(D_2) < cp(A)$, we can cut again Q_1^* and Q'_2 on $z_2 : D_2$ and obtain that

$Q_2^* : \ \Gamma - \{u : A\} \cup \Delta \cup \{z_3 : D_3, \ldots, z_n : D_n\}, \gamma_1[u/\delta] \cup \gamma_2[u/\delta] \cup \delta \cup \{z_2, \ldots, z_n\} \ \vdash^? \ z_n : q$

succeeds. by repeating this argument up to n (using lemma 5.4.8), we finally get that

$$Q_n^* : \ \Gamma - \{u : A\} \cup \Delta, \gamma_1[u/\delta] \cup \ldots \cup \gamma_1[u/\delta] \cup \delta \ \vdash^? \ z_n[z_n/x_n[u/y]] : q$$

succeeds. Since $x_n = x$ and $u \in \gamma$, we have

$$\gamma[u/\delta] = (\bigcup_{i=0}^{n} \gamma_i)[u/\delta] = \bigcup_{i=0}^{n} \gamma_i[u/\delta] \cup \delta.$$

Hence Q_n^* is Q^* and we are done. □

Corollary 5.4.10 (Modus Ponens) *If*

$\vdash^? \ \mathbf{v_0} : A \to B$ *and* $\vdash^? \ \mathbf{v_0} : A$ *succeed,*

then also

$\vdash^? \ \mathbf{v_0} : B$ *succeeds.*

Proof. If $\vdash^? \ \mathbf{v_0} : A \to B$ succeed, then also $Q = \ u : A, \{u\} \ \vdash^? \ u : B$ (with $u > \mathbf{v_0}$) succeeds by the implication rule. It easily seen that Q and $Q' = \ \ \vdash^? \ \mathbf{v_0} : A$ are compatible for cut, in every system. Thus, by cut we get that $\vdash^? \ \mathbf{v_0} : B$ succeeds. □

A general form of Cut

We have proved the admissibility of cut under specific compatibility conditions for each system. It is easily seen that we can always rename the labels in a query in order to match the compatibility constraints.

But we can do more: we can incorporate the needed re- labellings in the definition of cut in order to achieve a formulation which holds for every system without rstrictions. To this purpose, given $Q_1 = \Gamma[u : A], \gamma \ \vdash^? \ x : B$ and $Q_2 = \Delta, \delta \ \vdash^? \ y : A$ let us call the pair (ψ_1, ψ_2) of label substitutions *suitable for cutting* Q_1 *by* Q_2 *on* u if:

- ψ_1 and ψ_2 are order-preserving on $\gamma \cup \{u, x\}$ and on $\delta \cup \{y\}$, respectively[5];
- $pred(\gamma\psi_1, \psi_1(u)) < min(\delta\psi_2)$;
- $\psi_2(y) < min\{succ(\gamma\psi_1, \psi_1(u)), \psi_1(x)\}$.

Now the cut rule becomes

if $Q_1 = \Gamma[u : A], \gamma \ \vdash^? \ x : B$ and $Q_2 = \Delta, \delta \ \vdash^? \ y : A$ succeed

then also

$\Gamma\psi_1[\psi_1(u) : A/\Delta\psi_2], \gamma[\psi_1(u)/\delta\psi_2] \ \vdash^? \ \psi_1(x)[\psi_1(u)/\psi_2(y)] : B$ suceeds,

where (ψ_1, ψ_2) is any pair of label substitution *suitable for cutting* Q_1 *by* Q_2 *on* u. Given any query $Q = \Gamma, \gamma \ \vdash^? \ x : A$ and a label substitution ψ, $Q\psi$ is $\Gamma\psi, \gamma\psi \ \vdash^? \ \psi(x) : A$, thus the formula in the conclusion is nothing else that what is obtained by cutting (in the sense of theorem 5.4.9) $Q_1\psi_1$ by $Q_2\psi_2$ on $\psi_1(u)$. It is easy to see that this rule is admissible by theorem 5.4.9. We simply observe that if (ψ_1, ψ_2) are *suitable for cutting* Q_1 *by* Q_2 *on* u then

- Q_i succeeds implies $Q_i\psi_i$ succeeds for $i = 1, 2$;
- $COMP^S(Q_1\psi_1, Q_2\psi_2, \psi_1(u))$ holds for any system S.

Then the result follows by theorem 5.4.9.

5.5 Soundness and Completeness

We prove first the completeness of the proof procedure with respect to the Hilbert axiomatization of each system.

Definition 5.5.1 [Axiomatization of implication] We consider the following list of axioms:

1. $A \to A$;
2. $(B \to C) \to (A \to B) \to A \to C$;
3. $(A \to B) \to (B \to C) \to A \to C$;
4. $(A \to A \to B) \to A \to B$;
5. $(A \to B) \to ((A \to B) \to C) \to C$;
6. $A \to (A \to B) \to B$;
7. $A \to B \to A$.

Together with the following rules:

$$(MP) \qquad \frac{\vdash A \to B \quad \vdash A}{\vdash B}$$

[5]We say that a label substitution ψ is order-preserving on a set I if $z_1, z_2 \in I$ and $z_1 < z_2$ implies $\psi(z_1) < \psi(z_2)$.

Logic	Axioms
CL	(1), (2), (Suff)
T-W	(1), (2), (3)
T	(1), (2), (3), (4)
E-W	(1), (2), (3), (5)
E	(1), (2), (3), (4), (5)
L	(1), (2), (3), (6)
R	(1), (2), (3), (4), (6)
BCK	(1), (2), (3), (6), (7)

$$(Suff)\quad \frac{\vdash A \to B}{(B \to C) \to A \to C}$$

Each system is axiomatized by taking the closure under modus ponens (MP) and under substitution of the following combinations of axioms/rules

■

In the above axiomatization, we have not worried about the minimality and independence of the group of axioms for each system. For some systems the corresponding list of axioms given above is redundant, but it quickly shows some inclusion relations among the systems. We just remark that in presence of (5), (3) can be obtained by (2). Moreover, (5) is a weakening of (6). The rule of (Suff) is clearly obtainable from (3) and (MP). To have a complete picture we have included also intutionistic logic **I**. In figure **??** It is shown the lattice formed by these systems.

PUT A PROPER PICTURE

We prove that each system is complete with respect to the respective axiomatization. By corollary 5.4.10 and theorem 5.3.8, all we have to prove is that any atomic instance of each axiom succeeds in the relative proof- system. In addition, we need to show the admissibility of (Suff) rule for **CL**.

Theorem 5.5.2 (Completeness) *For every system S, if A is a theorem of S, then* $\vdash \mathbf{v_0} : A$ *succeeds in the corresponding proof system for S.*

Proof. We show a derivation of an arbitrary atomic instance of each axiom in the relative proof system. In the case of reduction, the condition $\gamma = \bigcup \gamma_i$, will not be explicitly shown, as its truth will be apparent by the choice of γ_i. Moreover, we assume that the truth of the condition for the case of immediate success is evident and we do not remark on that.

- (1) In all systems:
$$\vdash^? \ \mathbf{v_0} : a \to a$$
we step to
$$u : a, \{u\} \ \vdash^? \ u : a,$$
which immediately succeeds in all systems.

- (2) In all systems:
$$\vdash^? \ \mathbf{v_0} : (b \to c) \to (a \to b) \to a \to c.$$

three steps of the implication rule leads to:

$$\begin{array}{rcll} x_1 : b \to c, x_2 : a \to b, x_3 : a, \{x_1, x_2, x_3\} & \vdash^? & x_3 : c & \\ x_1 : b \to c, x_2 : a \to b, x_3 : a, \{x_2, x_3\} & \vdash^? & x_3 : b & \text{since } Red^S(\{x_1\}, \{x_2, x_3\}, x_1, x_3; x_3) \\ x_1 : b \to c, x_2 : a \to b, x_3 : a, \{x_3\} & \vdash^? & x_3 : a & \text{since } Red^S(\{x_2\}, \{x_3\}, x_2, x_3; x_3). \end{array}$$

- (3) In all systems, but **CL**:

$$\vdash^? \ \mathbf{v_0} : (a \to b) \to (b \to c) \to a \to c.$$

three steps of the implication rule leads to:

$$\begin{array}{rcll} x_1 : a \to b, x_2 : b \to c, x_3 : a, \{x_1, x_2, x_3\} & \vdash^? & x_3 : c & \\ (*)\ x_1 : a \to b, x_2 : b \to c, x_3 : a, \{x_1, x_3\} & \vdash^? & x_3 : b, & Red^S(\{x_2\}, \{x_1, x_3\}, x_2, x_3; x_3) \\ x_1 : a \to b, x_2 b \to c, x_3 : a, \{x_3\} & \vdash^? & x_3 : a, & Red^S(\{x_1\}, \{x_3\}, x_1, x_3; x_3). \end{array}$$

the step (*) is allowed in all systems, but those with (r4), namely **CL**.

- (4) In all systems, but those with (r1) or (r4):

$$\vdash^? \ \mathbf{v_0} : (a \to a \to b) \to a \to b.$$

two steps of the implication rule leads to:

$$x_1 : a \to a \to b, x_2 : a, \{x_1, x_2\} \vdash^? \ x_2 : b$$

By reduction we step to:

$$x_1 : a \to a \to b, x_2 : a, \gamma_1 \vdash^? \ x_2 : a \quad \text{and} \quad x_1 : a \to a \to b, x_2 : a, \gamma_2 \vdash^? \ x_2 : a$$

since $Red^S(\{x_1\}, \{x_2\}, \{x_2\}, x_1, x_2, x_2; x_2)$ holds in all systems without (r1) and (r4).

- (5) In all systems, but those with (r3) or (r4):

$$\vdash^? \ \mathbf{v_0} : (a \to b) \to ((a \to b) \to c) \to c.$$

two steps of the implication rule leads to:

$$\begin{array}{rcll} x_1 : a \to b, x_2 : (a \to b) \to c, \{x_1, x_2\} & \vdash^? & x_2 : c & \\ (*)\ x_1 : a \to b, x_2 : (a \to b) \to c, \{x_1\} & \vdash^? & x_2 : a \to b & Red^S(\{x_2\}, \{x_1\}, x_2, x_2; x_2) \\ x_1 : a \to b, x_2 : (a \to b) \to c, x_3 : a, \{x_1, x_3\} & \vdash^? & x_3 : b & \\ x_1 : a \to b, x_2 : (a \to b) \to c, x_3 : a, \{x_3\} & \vdash^? & x_3 : a & Red^S(\{x_2\}, \{x_1\}, x_2, x_2; x_2) \end{array}$$

The step (*) is allowed by (r2), but not by (r3) or (r4) since $max(\{x_1\}) = x_1 < x_2$.

- (6) In **L**,**R**,**BCK**:

$$\vdash^? \ \mathbf{v_0} : a \to (a \to b) \to b.$$

two steps of implication rule leads to:

$x_1 : a, x_2 : a \to b, \{x_1, x_2\} \vdash^? x_2 : b$
$x_1 : a, x_2 : a \to b, \{x_1\} \vdash^? x_1 : a$ since $Red^S(\{x_2\}, \{x_1\}, x_2, x_1; x_2)$.

- (7) In **BCK** we have:

$$\vdash^? \mathbf{v_0} : a \to b \to a.$$

two steps by implication rule leads to:

$$x_1 : a, x_2 : b, \{x_1, x_2\} \vdash^? x_2 : a$$

which succeeds by the success condition of **BCK**. In no other system this formula succeeds.

- As a last point, we prove the admissibility of (Suff) rule in **CL**. Let $\vdash^? \mathbf{v_0} : A \to B$ succeed. Then for any formula C, we have to show that

$\vdash^? \mathbf{v_0} : (B \to C) \to A \to C$ succeeds.

Let $C = C_1 \to \ldots \to C_n \to q$. Starting from

$\vdash^? \mathbf{v_0} : (B \to C_1 \to \ldots \to C_n \to q) \to A \to C_1 \to \ldots \to C_n \to q$

by the implication rule, we step to

$$\Delta, \{x_1, \ldots, x_{n+2} \vdash x_{n+2}\} : q,$$

where

$\Delta = \{x_1 : B \to C_1 \to \ldots \to C_n \to q, x_2 : A, x_3 : C_1, \ldots, x_{n+2} : C_n\}$

From the above query we step by reduction to:

$Q' = \Delta, \{x_2\} \vdash^? x_2 : B$ and $Q_i = \Delta\{x_{i+2}\} \vdash^? x_{i+2} : C_i$ for $i = 1, \ldots, n$.

since the condition for reduction are satisfied. By hypothesis, $\vdash^? \mathbf{v_0} : A \to B$ succeeds, which implies, by the implicational rule, that $x_2 : A, \{x_2\} \vdash^? x_2 : B$ succeeds, but then Q' succeeds by monotony. Queries Q_i succeed by proposition 5.3.3.

□

5.5.1 Soundness with respect to Fine's semantics

We prove now the soundness of the proof procedure with respect to a possible-world semantics. for substructural impicational logics. The first proposal of a possible world semantics is due to Urquhart. His proposal has been then elaborated by Fine, Routley, Meyer and Dozen in several directions. The intuitive idea of Urquhart semantics is to consider possible worlds equipped by a composition operation, i.e. forming an algebraic structure. We can consider the elements of the algebraic structure as resources which can be combined along a deduction. Urquhart considers the semilattice strucure as the underlying algerbaic structure. The semilattice semantics by Urquhart is a metalevel codification of the bookkeeping mechanism used to control natural deduction proofs in relevance logics. His semantical construction works well for the implicational fragment of a few systems, first of all **R**, but it cannot cover uniformly all the systems we are considering in this chapter. For instance in case of **E** we need consider two-dimensional models. Another problem of Urquharts semantics although we will not be affected in this section, is that this somantics does no longer work for the fragment of the language containing disjunction. The semantic we adopt here is a simplification of the one proposed by Kit Fine [] and elaborated more recently by Dozen[] [6]. There are, however, alternative semantics, as the one developed by Routley and Meyer based on a three-place accessibility relation on possible worlds. In the next section we will introduce it to prove the completeness of our procedures for a broader fragment. Although semantics is not our main concern, we observe that all these semantical variations are related; we refer to [] for more details.

Definition 5.5.3 Let us fix a language $\mathcal{L}$, a structure M is a tuple of the form:

$$M = (W, \leq, \circ, 0, V),$$

where W is a non empty set, $\circ$ is a binary operation on S, $0 \in S$, $\leq$ is a partial order relation on W, V is a function of type $W \to Pow(Var)$. In all structures the following properties are assumed to hold:

$0 \circ x = x$,
$x \leq y$ implies $x \circ z \leq y \circ z$,
$x \leq y$ implies $V(x) \subseteq V(y)$.

The structures may satisfy the following conditions:

(1) $x \circ (y \circ z) \leq (x \circ y) \circ z$;
(2) $x \circ (y \circ z) \leq (y \circ x) \circ z$;
(3) $x \circ (x \circ y) \leq x \circ y$;
(4) $(x \circ y) \circ y \leq x \circ y$;
(5) $x \circ x \leq x$;
(6) $x \circ 0 \leq x$;
(7) $x \circ y \leq y \circ x$;
(8) $0 \leq x$.

Truth conditions for $x \in W$, we define

- $M, x \models p$ if $p \in V(x)$;
- $M, x \models A \to B$ if

$$\forall y \in W(M, y \models A \;\Rightarrow\; M, x \circ y \models B).$$

We say that A is *true* in M (denoted by $M \models A$) if $M, 0 \models A$.

Each logic is characterized by a class of structures which satisfy some of the above conditions. The mapping of logics– group of conditions is shown in figure 5.5.3. We have included also intutionistic logic **I** to show its proper place within this framework. As in the axiomatization, the group of conditions for each system is deliberately redundant, it is given in this way to quickly show the inclusion relations of the systems. For instance, given (5) and (7), the condition (3) and (4) immediately follows.

■

We assume that $\circ$ associates to the left, that is we write:

[6]Dealing only with the implicational fragment, we have simplified Fine's semantics: we do not have *prime* or maximal elements. The main difference with Dozen's grupoids models is that we do not assume an underlying semilattice structure.

Logic	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
CL								
T-W	*	*						
T	*	*	*	*	*			
E-W	*	*				*		
E	*	*	*	*	*	*		
L	*	*				*	*	
R	*	*	*	*	*	*	*	
BCK	*	*				*	*	*
I	*	*	*	*	*	*	*	*

$x \circ y \circ z = (x \circ y) \circ z.$

The following lemma, (whose proof is left to the reader), shows that the hereditary property extends to all formulas.

Lemma 5.5.4 *Let $M = (S, \leq, \circ, 0, V)$ be an S-structure, let $x, y \in S$, then for any formula A:*

if $M, x \models A$ and $x \leq y$, then $M, y \models A$.

The axiomatization of definition 5.5.1 is sound and complete with respect to this semantics.

Theorem 5.5.5 *[REFERENCES????] $\models_S A$ if and oly if A is derivable in the corrsponding axiom system of Definition 5.5.1.*

In order to proof the soundness of our procedure, we need to interpret databases and queries in this semantics. As usual, we introduce the notion of realization of a database.

Definition 5.5.6 [Realization] For each system S, given a database Γ, and a set of resources of γ, an *S-realization* of $\Gamma, \gamma)$ in an S-structure $M = (S, \circ, \leq, 0, V)$, is a mapping

$\rho : \mathcal{A} \rightarrow S$

such that:

1. $\rho(\mathbf{v_0}) = 0$;
2. if $y : B \in \Gamma$ then $M, \rho(y) \models B$.

■

In order to define the notion of validity of a query, we need to introduce some further notation. Given an S-realization ρ, γ and x, we define

$\rho(\gamma) = 0$ if $\gamma = \emptyset$,
$\rho(\gamma) = \rho(x_1) \circ \ldots \circ \rho(x_n)$ if $\gamma = \{x_1, \ldots, x_n\}$, where $x_1 < \ldots < x_n$
$\rho(< \gamma, x >) = \rho(\gamma)$ if $x \in \gamma$,
$\rho(< \gamma, x >) = \rho(\gamma) \circ 0$ if $x \notin \gamma$.

Moreover, if $\gamma = \{x_1, \ldots, x_n\}$, $\delta = \{y_1, \ldots, y_m\}$ (ordered as shown) we write

$$\rho(\gamma) \circ \rho(\delta) = \rho(x_1) \circ \ldots \circ \rho(x_n) \circ (\rho(y_1) \circ \ldots \circ \rho(y_m))$$

Definition 5.5.7 [Valid query] Let $Q = \Gamma, \gamma \vdash^? x : A$ be an S-regular query, we say that Q is *S-valid* if for every S-structure M, for every realization ρ of Γ in M, we have

$$M, \rho(< \gamma, x >) \models A.$$

■

We prove the soundness of the proof-systems with respect to the semantics we have just introduced. To this purpose, we need to relate the conditions involved in the reduction rule to the algebraic properties of $\circ$ as defined above. This is the purpose of the following lemma.

Lemma 5.5.8 *Given an S-regular query $\Delta, \delta \vdash^? x : G$, let δ_i and x_i, for $i = 0, \ldots, k$, such that*

1. $\delta_0 = \{z\}$, $x_0 = z$, *for some* $z \in \delta$
2. $\bigcup_{i=0}^{k} \delta_i = \delta$.
3. $Red^S(\delta_0, \ldots, \delta_k, x_0, \ldots, x_k; x)$

For every S-structure M and every realization ρ of (Δ, δ) in M, we have

$$\rho(< \delta_0, x_0 >) \circ \rho(< \delta_1, x_1 >) \circ \ldots \circ \rho(\delta_k, x_k >) \leq \rho(< \delta, x >)$$

Proof. Each system S requires a separate consideration.

The easiest cases are those of **R**, **L**, and **BCK**. By regularity, we have that $x \in \delta_i$, and $x \in \delta$, which implies that $\rho(< \delta_i, x_i >) = \rho(\delta_i)$ and $\rho(< \delta, x >) = \rho(\delta)$. In the cases of **R**, **L**, **BCK**, $(S, \circ, 0)$ is a commutative monoid, thus the order of composition of elements does not matter. Hence, for **L** and **BCK**, the conditions $\bigcup_{i=0}^{k} \delta_i = \delta$ and $\delta_i \cap \delta_j = \emptyset$ imply that

$$\rho(\delta_0) \circ \rho(\delta_1) \circ \ldots \circ \rho(\delta_k) = \rho(\delta).$$

In case of **R**, we can regard $\rho(\alpha)$, for any α, as denoting a multiset, then the condition $\bigcup_{i=0}^{k} \delta_i = \delta$ imply that

$$(*)\ \rho(\gamma) \subseteq\mid \rho(\gamma_0) \circ \ldots \circ \rho(\gamma_k),$$

(considering $\circ$ as multiset union). But (*), by condition (5) implies that

$$\rho(\delta_0) \circ \rho(\delta_1) \circ \ldots \circ \rho(\delta_k) \leq \rho(\delta).$$

With regard to the other cases, we give a proof of the claim for **E**, the other cases are similar, but simpler. We prove that there are $\beta_0, \ldots, \beta_k$, such that $\beta_0 = \delta_0$, $\beta_k = \delta$, and for $i = 1, \ldots, k$, it holds that:

$$(*)\ \rho(\beta_{i-1}) \circ \rho(< \delta_i, x_i >) \leq \rho(\beta_i).$$

To this purpose we need the notion of ordered merge of two resource sets α, β. We can consider α and β as ordered sequence of labels without repetitions. We use the notation $\alpha * x$ to denote the sequence α followed by x, whenever x does not occur in α and $max(\alpha) < x$. We denote by ϵ the empty sequence. The ordered merge of α, β, denoted by $mo(\alpha, \beta)$ is defined as follows:

$mo(\alpha, \epsilon) = mo(\emptyset, \alpha) = \alpha$,
$mo(\alpha * x, \beta * y) = mo(\alpha, \beta * y) * x$ if $y < x$,
$mo(\alpha * x, \beta * y) = mo(\alpha * x, \beta) * y$ if $x < y$,
$mo(\alpha * x, \beta * y) = mo(\alpha, \beta) * x$ if $x = y$.

Going back to what we have to show, we let $\beta_i = mo(\beta_{i-1}, \delta_i)$ and we split the proof of (*) into two cases according to whether $x_i \in \delta_i$ or not.

(1) If $x_i \in \delta_i$, then we prove that $\rho(\beta_{i-1}) \circ \rho(\delta_i) \leq \rho(\beta_i)$.

(2) If $x_i \notin \delta_i$, then we prove that $\rho(\beta_{i-1}) \circ (\rho(\delta_i) \circ 0) \leq \rho(\beta_i)$.

- Case (1). It is easy to see that $max(\beta_{i-1}) \leq x_i$, hence we get $max(\beta_{i-1}) \leq max(\delta_i)$. We prove the claim by induction on $|\beta_{i-1}| + |\delta_i|$. Assume both β_{i-1} and δ_i are $\neq \epsilon$ (otherwise the argument below simplifies), so that $\beta_{i-1} = \beta' * u$ and $\delta_i = \delta' * x_i$.

  (Subcase a) $u = x_i$, then we have:

$$\begin{array}{rcl} ro(\beta' * x_i) \circ \rho(\delta' * x_i) & = & \rho(\beta') \circ \rho(x_i) \circ (\rho(\delta') \circ \rho(x_i)) \\ & \leq & (\rho(\delta') \circ (\rho(\beta') \circ \rho(x_i)) \circ \rho(x_i)) \\ & \leq & \rho(mo(\delta', \beta' * x_i) \circ \rho(x_i), \text{ by induction hypothesis} \\ & = & (\rho(mo(\delta', \beta') \circ \rho(x_i)) \circ \rho(x_i) \\ & \leq & \rho(mo(\delta', \beta') \circ \rho(x_i) \text{ by property (4)}^7 \\ & = & \rho(mo(\beta' * x_i, \delta' * x_i). \end{array}$$

  (Subcase b) let $max(\beta') = u < x_i$. We have two further subcases, according to $max(\gamma') \leq u$ or $max(\gamma') > u$. In the former case we proceede similarly to (subcase a). In the latter case, we have:

$$\begin{array}{rcl} \rho(\beta' * u) \circ \rho(\delta' * x_i) & \leq & (\rho(\beta') \circ \rho(u) \circ (\rho(\delta')) \circ \rho(x_i)) \\ & \leq & \rho(mo(\beta' * u, \delta')) \circ \rho(x_i) \text{ by induction hypothesis} \\ & = & \rho(mo(\beta' * u, \delta')) \circ \rho(x_i). \end{array}$$

- Case (2) If $\beta_{i-1} = \epsilon$, then we have $\rho(\beta_{i-1}) = 0$, whence

$$\begin{array}{rcl} \rho(\beta_{i-1}) \circ (\rho(\delta_i) \circ 0) & = & 0 \circ (\rho(\delta_i) \circ 0) \\ & = & \rho(\delta_i) \circ 0 \\ & \leq & \rho(\delta_i) \text{ by property (6)} \\ & = & \rho(mo(\beta_{i-1}, \beta_i)). \end{array}$$

  If $\beta \neq \epsilon$, let $\beta_{i-1} = \beta' * u$; we have two cases: if $u \leq max(\delta_i)$, then

$$\begin{array}{rcl} \rho(\beta' * u) \circ (\rho(\delta_i) \circ 0) & \leq & (\rho(\beta') \circ \rho(u) \circ (\rho(\delta_i)) \circ 0 \\ & \leq & \rho(\beta') \circ \rho(u) \circ (\rho(\delta_i) \text{ by property (6)} \end{array}$$

  then we can conclude by case (1).

  If $max(\delta_i) < u$, then

$$\begin{array}{rcl} \rho(\beta' * u) \circ (\rho(\delta_i) \circ 0) & \leq & (\rho(\delta_i) \circ (\rho(\beta') \circ \rho(u))) \circ 0 \\ & \leq & \rho(\delta_i) \circ (\rho(\beta') \circ \rho(u)), \text{ by property (6)} \end{array}$$

  again we can conclude by case (1).

□

Theorem 5.5.9 *Let $Q = \Gamma, \gamma \vdash^? x : A$ be an S-regular query, if Q succeeds in the proof-system S, then Q is S-valid.*

Proof. By induction on the height h of a successful derivation of Q.

- If $h = 0$, then the query immediately succeeds, $x : q \in \Gamma$ and $Succ^S(\gamma, x)$ holds. Let M be an S-structure and ρ an S-realization of Γ in M. We have $M, \rho(x) \models q$. In case of **BCK**, we have that $x \in \gamma$; it easily seen that $\rho(x) \leq \rho(<\gamma, x>)$, whence the result follows by the hereditary property; in all the other cases, we have $\gamma = \{x\}$, thus the claim follows by $\rho(x) = \rho(<\gamma, x>)$.

- Let $h > 0$ and the implication rule is applied to derive Q, that is, $A = B \to C$ and from Q, we step to Q':

  $$Q' = \Gamma \cup \{z : B\}, \gamma \cup \{z\} \vdash^? z : C,$$

  where $z > max(Lab(\Gamma)$. Q' succeeds by a derivation of smaller height. Suppose that Q is not S-valid, let M be an S-structure and ρ a realization of Γ, such that $M, \rho(<\gamma, x>) \not\models B \to C$. Let $\rho(<\gamma, x>) = a$, then there is some $b \in S$, such that $M, b \models B$, but $M, a \circ b \not\models C$. Let $Q' = \Gamma' = \Gamma \cup \{z : C\}$, $\gamma' = \gamma \cup \{z\}$. By induction hypothesis Q' is S-valid. We can define a realization ρ' for , Γ', by letting $\rho'(z) = b$, and $\rho'(u) = \rho(u)$, for $u \neq z$. Notice that

  $$\rho'(\gamma') = \rho(\gamma) \circ \rho'(z) = a \circ b.$$

  We have that

  $$M, \rho'(\gamma') \not\models C,$$

  which contradicts the induction hypothesis on Q'.

- Let $h > 0$ and suppose the first step in a derivation of Q is by reduction. Then A is an atom q, there is $z : C \in \Gamma$, with $C : D_1 \to \ldots \to D_k \to q$, and from Q we step to

  $$Q_i : \Gamma, \gamma_i \vdash^? x_i : D_i,$$

  for some δ_i, and x_i for $i = 0, \ldots, k$ such that:

  1. $x_0 = z$, $\gamma_0 = \{z\}$, and $\bigcup_{i=0}^{k} \gamma_i = \gamma$.
  2. $Red^S(\gamma_0, \ldots, \gamma_k, x_0, \ldots, x_k; x)$

  Let M be an S-structure and ρ be a realization of Γ in M. We show that $M, \rho(<\gamma, x>) \models q$ holds. Since each Q_i succeeds by a derivation of smaller height, by induction hypothesis, we get that

  $$M, \rho(<\gamma_i, x_i>) \models D_i.$$

  On the other hand, being ρ a realization of Γ we have:

  $$M, \rho(<\gamma_0, x_0>) \models D_1 \to \ldots \to D_k \to q.$$

  Thus, we obtain that

$$M, \rho(<\gamma_0, x_0>) \circ \rho(<\gamma_1, x_1>) \circ \ldots \circ \rho(<\gamma_k, x_k>) \models q.$$

By lemma 5.5.8, we have

$$\rho(<\gamma_0, x_0>) \circ \rho(<\gamma_1, x_1>) \circ \ldots \circ \rho(\gamma_k, x_k>) \leq \rho(<\gamma, x>),$$

Thus, by the hereditary property we obtain:

$$M, \rho(<\gamma, x>) \models q.$$

□

5.6 Extending the language to Harrop formulas

In this section we show how we can extend the language by other connectives. We expand our language by allowing certain occurrences of lattice or extensional conjunction ($\wedge$) , disjunction ($\vee$) and intensional conjunction, called tensor, ($\otimes$). The distinction between $\wedge$ and $\otimes$ is typical of substructural logics and it comes from the rejection of some of the structural rules: $\wedge$ is the usual lattice connective, $\otimes$ is the residual operator with respect to $\rightarrow$[8]. This fragment might be interesting as a base of a family of a sort of logic-programming languages based on substructural logics.

This connectives are governed by the following axioms and rules (we take the axiomatization by Dozen [],[]).

Definition 5.6.1 [Axioms for $\wedge, \otimes, \vee$]

- ($\otimes$)

  (1) $((A \rightarrow B) \otimes A) \rightarrow B$,
  (2) $A \rightarrow (B \rightarrow (A \otimes B))$,
  (3) $\dfrac{\vdash A \rightarrow A_1 \quad \vdash B \rightarrow B_1}{\vdash (A \otimes B) \rightarrow (A_1 \otimes B_1)}$.

- ($\wedge$)

  (1) $A \wedge B \rightarrow A$,
  (2) $A \wedge B \rightarrow B)$,
  (3) $\dfrac{\vdash C \rightarrow A \quad C \rightarrow B}{\vdash C \rightarrow A \wedge B}$.

- ($\vee$)

  (1) $A \rightarrow A \vee B$,
  (2) $B \rightarrow A \vee B$,
  (3) $\dfrac{\vdash A \rightarrow D \quad \vdash B \rightarrow D}{\vdash A \vee B \rightarrow D}$,
  (4) $\dfrac{\vdash C \rightarrow A \rightarrow D \quad \vdash C \rightarrow B \rightarrow D}{\vdash C \rightarrow (A \vee B) \rightarrow D}$.

We will however limit our consideration to a frament of the language, defined below. We justify the need of this restriction immediately after. As usual we distinguish D-formula which are the constituents of databases and G-formulas which can be asked as goals. ■

[8]An alternative terminology which has become popular after Girard's linear logic is: $\wedge$ is the *addittive* conjunction and $\otimes$ is the *multiplicative* conjunction.

Definition 5.6.2 We define D-formulas and G-formulas as follows:

$D := q \mid G \rightarrow q,$
$CD := D \mid CD \wedge CD;$
$G := q \mid G \wedge G \mid G \vee G \mid G \otimes G \mid CD \rightarrow G.$

A database Δ is a finite set of *labelled sets* of D- formulas.

■

Notice that CD formulas are just conjunction of D-formulas.

The idea of the previous definition is that a database corresponds to a $\otimes$ -conjunction of $\wedge$-conjunction of D-formulas. Notice that each D-formula either is atomic, or is an implication, or is a $\wedge$-conjunction of D-formulas. Within a database, CD formulas are therefore decomposed into sets of D-formulas. We have imposed another (inessential) restriction on D-formulas: we do not allow D-formulas of the type:

$G_1 \rightarrow G_2 \rightarrow q$, where G_1 and G_2 are G-formulas.

But the above formula can be rewritten as

$G_1 \otimes G_2 \rightarrow q$

which is accepted.

The extent of this fragment is shown by the following propositions.

Lemma 5.6.3 *Every formula on $(\wedge, \vee, \rightarrow, \otimes)$ without positive occurrences of $\otimes$ and $\vee$ and without occurrences of $\otimes$ within a negative occurrence of $\wedge$ is equivalent to a conjunction (set) of D-formulas.*

We have a dual result for G-formulas. The reason why we have put the restriction on nested occurrences of $\otimes$ within $\wedge$ is that, on the one hand, we want to keep the simple labelling mechanism we have used for the implicational fragment, on the other hand we want to identify a common fragment for all systems to which the computation rules are easily extended. The labelling mechanism does no longer work if we relax this restriction. For instance, how could we handle $A \wedge (B \otimes C)$ as a D-formula? We should add $x : A$ and $x : B \otimes C$, in the database. The formula $x : B \otimes C$ cannot be decomposed, unless we use complex labels: intuitively we should split x into some y and z, add $y : B$ and $z : C$, and remember that x, y, z are connected (in terms of Fine's semantics the connection would be expressed as $x = y \circ z$). We prefer to accept the above limitation and postpone the investigation of a broader fragment to future work [9].

The computation rules can be extended to this fragment without a great effort.

Definition 5.6.4 [Proof system for the extended language] A query has the form

$\Delta, \delta \vdash^? x : G,$

where Δ is a set of D-formulas and G is a G-formula.

- (success)

[9]In some logics, such as **L**, we don't need this restriction since we have: $\Gamma, A \wedge B \vdash C$ implies $\Gamma, A \vdash C$ or $\Gamma, B \vdash C$. Thus, we can avoid to introduce extensional conjunctions atall into the database, by introducing only one of the two (at choice), this approach is followed by Harland []. Howevern this property does not hold for **R** and other logics

$$\Delta, \delta \vdash^? x : q; \text{ succeeds},$$

if $x; q \in \Delta$ and $Succ^S(\delta, x)$.

- (implication) from

  $$\Delta, \delta \vdash^? x : CD \to G$$

  if $CD = D_1 \wedge \ldots \wedge D_n$, we step to

  $$\Delta \cup \{y : D_1, \ldots, y : D_n\}, \delta \cup \{y\} \vdash^? y : G$$

  where $y > max(Lab(\Delta))$, (hence $y \notin Lab(\Delta)$);

- (reduction) from

  $$\Delta, \delta \vdash^? x : q$$

  if there is $z : G' \to q \in \Delta$ and there are δ_1, x_1 such that:

  1. $\delta = \delta_1 \cup \{z\}$.
  2. $Red^S(\{z\}, \delta_1, z, x_1; x)$

  then we step to

  $$\Delta, \delta_1, \vdash^? x_1 : G'.$$

- (conjunction) from

  $$\Delta, \delta \vdash^? x : G_1 \wedge G_2$$

  step to

  $$\Delta, \delta \vdash^? x : G_1 \text{ and } \Delta, \delta \vdash^? x : G_2.$$

- (disjunction) from

  $$\Delta, \delta \vdash^? x : G_1 \vee G_2$$

  step to

  $$\Delta, \delta \vdash^? x : G_i \text{ for } i = 1 \text{ or } i = 2.$$

- (tensor) from

  $$\Delta, \delta \vdash^? x : G_1 \otimes G_2$$

  if there are δ_1, δ_2, x_1 and x_2 such that

  1. $\delta = \delta_1 \cup \delta_2$,
  2. $Red^S(\delta_1, \delta_2, x_1, x_2; x)$,

  step to

  $$\Delta, \delta_1 \vdash^? x_1 : G_1 \text{ and } \Delta, \delta_2 \vdash^? x_2 : G_2.$$

■

Notice that in case of reduction, the conditions grouped in the predicate Red^S, become as follows: $\delta = \delta_1 \cup \{z\}$ and

1. $z \notin \delta_1$,
2. $\delta_1 \neq \emptyset$,
3. $z \leq x_1 \wedge z_1 = max(\delta_1)$,
4. $z \leq x_1 \wedge z_1 \geq max(\delta_1)$,
5. $z < min(\delta_1) \wedge x_1 = max(\delta_1)$.

Example 5.6.5 Let Δ be the following database:

$x_1 : e \wedge g \rightarrow d$,
$x_2 : (c \rightarrow d) \otimes (a \vee b) \rightarrow p$,
$x_3 : c \rightarrow e$,
$x_3 : c \rightarrow g$,
$x_4 : (c \rightarrow g) \rightarrow b$.

In figure 5.1, we show a successful derivation of

$$\Delta, \{x_1, x_2, x_3, x_4\} \vdash^? x_4 : p,$$

in relevance logic **E** (and stronger systems). The success of this query corresponds to the validity of the following formula in **E**:

$$\big[(e \wedge g \rightarrow d) \otimes ((c \rightarrow d) \otimes (a \vee b) \rightarrow p) \otimes [(c \rightarrow e) \wedge (c \rightarrow g)] \otimes ((c \rightarrow g) \rightarrow b)\big] \rightarrow p.$$

■

We turn now to prove the soundness and the completeness of the procedure for the extended language. To this purpose we will use the alternative, but related, semantics for substructural logics elaborated by Routley and Meyer. It is a Kripke-possible world semantics which makes use of a three-place accessibility relation on possible worlds. The intuitive meaning of this ternary relation $Rxyz$ has been subject to discussion. If we interpret x, y, z as pieces of information, one can read $Rxyz$ as something of the sort 'the combination of x and y is included in z'. With this interpretation, there is an obvious connection with Fine and Dozen semantics presented in section **??**. Given $x, y, z \in W$, we can define the ternary relation $x \circ y \leq z$. Thus, every S-structure determine a Routley-Meyer structure. But not vice versa.

Another reading of $Rxyz$ is relative inclusion:

$y \leq_x z$ 'y is included in z, from the point of view of x.

The reason why we introduce Routley-Meyer semantics is simply that it is easier to proof the completeness (via a canonical model construction) with respect to this semantics, rather than with respect to Fine's model as defined in sextion **??**. Moroever, the correspondence of our deduction procedure with the Routley-Meyer semantics might be of interest by itself.

$$\begin{tikzcd}[column sep=tiny]
 & & \Delta, \{x_1, x_2, x_3, x_4\} \vdash^? x_4 : p \arrow[d, dash] & \\
 & & \Delta, \{x_1, x_3, x_4\} \vdash^? x_4 : (c \to d) \otimes (a \vee b) \arrow[dl, dash] \arrow[dr, dash] & \\
 & \Delta, \{x_1, x_3\} \vdash^? x_3 : c \to d \arrow[d, dash] & & \Delta, \{x_3, x_4\} \vdash^? x_4 : a \vee b \arrow[d, dash] \\
 & \Delta, x_5 : c, \{x_1, x_3, x_5\} \vdash^? x_5 : d \arrow[d, dash] & & \Delta, \{x_3, x_4\} \vdash^? x_4 : b \arrow[d, dash] \\
 & \Delta, x_5 : c, \{x_3, x_5\} \vdash^? x_5 : e \wedge g \arrow[dl, dash] \arrow[dr, dash] & & \Delta, \{x_3\} \vdash^? x_4 : c \to g \arrow[d, dash] \\
\Delta, x_5 : c, \{x_3, x_5\} \vdash^? x_5 : e \arrow[d, dash] & & \Delta, x_5 : c, \{x_3, x_5\} \vdash^? x_5 : g \arrow[d, dash] & \Delta, x_6 : c, \{x_3, x_6\} \vdash^? x_6 : g \arrow[d, dash] \\
\Delta, x_5 : c\{x_5\} \vdash^? x_5 : c & & \Delta, x_5 : c, \{x_5\} \vdash^? x_5 : c & \Delta, x_6 : c, \{x_6\} \vdash^? x_6 : c
\end{tikzcd}$$

Figure 5.1:

Definition 5.6.6 Let us fix a language $\mathcal{L}$, an S-structure M for $\mathcal{L}$ is a tuple of the form:

$$M = (W, R, 0, V),$$

where W is a non empty set, R is a ternary relation on W, $0 \in W$, V is a function of type $W \to Pow(Var)$. We will define also the relation $\leq$ by stipulating:

$a \leq b \equiv R0ab.$

In all structures the following properties are assumed to hold:

(i) $\leq$ is transitive and reflexive.
(ii) $x' \leq x$ and $Rxyz$ implies $Rx'yz$
(iii) $x \leq y$ implies $V(x) \subseteq V(y)$.

The structures may satisfy additional properties. To this purpose we introduce some more notation.

$R^2(xy)zu \equiv \exists a(Rxya \wedge Razu)$
$R^2x(yz)u \equiv \exists a(Rxau \wedge Ryza)$.

The conditions are the following:

(1) $R^2(xy)zu \to R^2x(yz)u$
(2) $R^2(xy)zu \to R^2y(xz)u$
(3) $Rxyz \to R^2(xy)yz$
(4) $\forall x \exists y(Rxyx \wedge \forall u \forall v(Ryuv \to R0uv))$
(5) $Rxyz \to Ryxz$.
(6) $R00a$.

Truth conditions for $x \in W$, we define

Logic	(1)	(2)	(3)	(4)	(5)	(6)
CL	*					
T-W	*	*				
T	*	*	*			
E-W	*	*		*		
E	*	*	*	*		
L	*	*		*	*	
R	*	*	*	*	*	
BCK	*	*		*	*	*

- $M, x \models p$ if $p \in V(x)$;
- $M, x \models A \to B$ if
$$\forall y, z \in W(Rxyz \wedge M, y \models A \ \Rightarrow \ M, z \models B).$$
- $M, x \models A \wedge B$ iff $M, x \models A$ and $M, x \models B$;
- $M, x \models A \otimes B$ iff there are $x, y \in S$, such that $Ryzx$ and $M, y \models A$ and $M, z \models B$;
- $M, x \models A \vee B$ iff $M, x \models A$ or $M, x \models B$.

We say that A is *true* in M (denoted by $M \models A$) if $M, 0 \models A$. We say that A is S-valid if for all S-structures $M, 0 \models A$ and it is denoted by $\models_S A$.

The mapping between logic and conditions is shown in Figure 5.6.6.

■

The proof of the soundness is similar to the one of theorem 5.5.5, namely one proceeds by induction on the length of computations. Alternatively, one can extend the proof of the soundness with respect to Fine's semantics of theorem 5.5.9 to this fragment. An then, one can rely upon the equivalnce of Routley-Meyer semantics and Fine semantics for this fragement (see [**?**] for details[10]).

Theorem 5.6.7 *Let $\Gamma, \gamma \vdash^? x : G$ succeeds in the system S, $\gamma = \{x_1, \ldots, x_k\}$ (ordered as shown) and $i = 1, \ldots, k$ let $S_i = \{A \mid x_i : A \in \Gamma\}$; then*

$$\models_S (\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \to G.$$

In order to prove the completeness, we need to ensure some closure under cut. Since we have made a distinction between D-formulas and G-formulas, we must be pay attention to what formulas we cut. In general, one can prove the cut admissibility property for formulas which are *simultaneously* G and D-formulas. But this is not enough the completeness. What we need is a sort of 'closure under cut' with respect to D-formulas. The point is that our proof-procedure cannot prove D-formulas. Thus, we must express it rather indirectly. The property is given in the next proposition and its meaning will be apparent in the completeness proof.

[10]When disjunction is present the simplified Fine semantics of sectionfinesem cannot longer be adopted. Models have to be enriched by a subset of special worlds which correponds to 'prime' theories. Fine shows that every Rutley-Meyer model which satisfy the additional property $Rxyz \wedge z \leq u \to Rxyu$ determines a Fine model with the enriched structure. In our case, we can assume that Routley-Meyer models satisfy this further property without loss of generality. This is shown by the proof of the completeness theorem, namely the canonical model built in that proof satisfies this condition.

The cut theorem required a compatibility constraint on the queries which was denoted by $Comp^S$. We need it again. It can be seen that this predicate only depends on the labels, but not on the formulas of the involved databases, so that given $\Gamma, \gamma, x, u \in Lab(\Gamma)$, where u is the position of the cut-formula and x the current position on Γ, and similarly, Δ, δ, $y \in Lab(\Delta)$, we can write $Comp^S(\Gamma, \gamma, x, u; \Delta, \delta, y)$ to denote the same set of conditions as in definition 5.4.2. This is an abuse of notation, but it is necessary, as we do not have *two* queries to cut.

Proposition 5.6.8 *Let (1)* $\Gamma[u : D_1, \ldots, u : D_k], \gamma \vdash^? x : G$ *succeed. Let* Δ, δ, y *such that the following condtion holds*

1. $Comp^S(\Gamma, \gamma, x, u; \Delta, \delta, y)$,
2. *for* $i = 1, \ldots, k$,
   - *if* $D_i = q_i$ *atomic, then* $\Delta, \delta \vdash^? y : q_i$ *succeeds;*
   - *if* $D_i = G_i \rightarrow q_i$, *then for all* Σ, σ, z *such that* $Red^S(\delta, \sigma, y, z; z)$ *and* $\Sigma, \sigma \vdash^? z : G_i$ *succeeds,* $\Delta \cup \Sigma, \delta \cup \sigma \vdash^? z : q_i$ *succeeds.*

Then $\Gamma[u : D_1, \ldots, u : D_k/\Delta], \gamma[u/\delta] \vdash^? x[u/y] : G$ *succeeds.*

Proof. By induction on the length of a derivation of (1). □

The completeness can now be proved by a canonical model construction.

Definition 5.6.9 Let $M = (W, R, 0, V)$ be the structure defined as follows:

- $0 = (\emptyset, \mathbf{v_0})$,
- W is the set of pairs (Γ, x) (including 0) such that Γ is a database and x is a label, with the following restrictions:

  -for **R**, **L**, **BCK**, if $\Gamma \neq \emptyset$, then $x \in Lab(\Gamma)$,
  -for **T**, **T-W**, **CL**, $x = max(Lab(\Gamma))$ (thus, if $\Gamma = \emptyset$, then $x = \mathbf{v_0}$),
  -for **E**, **E-W**, $x \geq max(Lab(\Gamma))$.

- The relation R

  $R(\Gamma, x)(\Delta, y)(\Sigma, z)$

  holds iff the following conditions (1) and (2) (or (1) and (2') for **BCK**) hold

  (1) $Red^S(\lambda_\Gamma, \lambda_\Delta, x, y; z)$
  (2) $\Gamma \cup \Delta = \Sigma$ for all systems, but **BCK**,
  (2')$\Gamma \cup \Delta \subseteq \Sigma$ for **BCK**.

- $V(\Gamma, x) = \{p :$ atom $\mid \Gamma, \lambda_\Gamma \vdash^? x : p$ succeeds$\}$.

■

We now prove that the structure we have just defined satisfies all properties of definition 5.6.9.

Proposition 5.6.10 *The relation* $(\Gamma, x) \leq (\Delta, y) \equiv R0(\Gamma, x)(\Delta, y)$ *satisfies the conditions (i)-(iii) of definition 5.6.9.*

Proof. For condition (i), i.e. $\leq$ is a reflexive and transitive relation, we use the fact that for every (Γ, x), we have $Red^S(\emptyset, \lambda_\Gamma, \mathbf{v_0}, x; x)$. Condition (ii) is easy too, since $(\Gamma, x) \leq (\Delta, y)$ implies $\lambda_\Gamma = \lambda_\Delta$, (or $\lambda_\Gamma \subseteq \lambda_\Delta$ for **BCK**) and $x = y$, in the cases where it matters (i.e. all cases, except **L**, **R**, **BCK**). We get that, given any (Σ, z),

$Red^S(\lambda_\Delta, \lambda_\Sigma, y, z; z)$ implies $Red^S(\lambda_\Gamma, \lambda_\Sigma, x, z; z)$.

From this (ii) easily follows. For (iii), let $\Gamma, \lambda_\Gamma \ \vdash^? \ x : p$ succeed and $(\Gamma, x) \leq (\Gamma, y)$. In case of **T**, **T-W** and **CL** we have that $\Gamma = \Delta$ and $x = y$, thus the result is obious. In all other cases, we conclude by propositions 5.3.4 and 5.3.5. □

Proposition 5.6.11 *For each system S, the relation R satisfies the specific conditions for S.*

Proof. One has to check all properties for each specific system. As an example, we check properties (1) and (4). The others are similar and left to the reader.

For **(1)**, let $R^2((\Gamma, x)(\Delta, y))(\Sigma, z)(\Pi, u)$ (We omit the indication of S from R^S). Then there is (Φ, v) such that

$R(\Gamma, x)(\Delta, y)(\Phi, v)$ and $R(\Phi, v)(\Sigma, z)(\Pi, u)$.

We have to show that there is (Ψ, w) such that

$R(\Delta, y)(\Sigma, z)(\Psi, w)$ and $R(\Gamma, x)(\Psi, w)(\Pi, u)$.

Among the several cases, we consider the one of **E** and **E-W**, the others are similar, perhaps simpler. By hypothesis we have

$\Gamma \cup \Delta = \Phi$ and $\Phi \cup \Sigma = \Pi$,
$Red(\lambda_\Gamma, \lambda_\Delta, x, y; v)$, $Red(\lambda_\Phi, \lambda_\Sigma, v, z; u)$, which implies $y = v$ and $z = u$.

Let us take $(\Psi, w) = (\Delta \cup \Sigma, z)$. Since $Red(\lambda_\Phi, \lambda_\Sigma, v, z, u)$, we have $v \leq z$, whence $y \leq z$. Thus $max(\lambda_\Psi) \leq z$ and (Ψ, z) is well- defined. Clearly,

(*) $\Delta \cup \Sigma = \Psi$ and
(**) $\Gamma \cup \Psi = \Pi$.

From $Red(\lambda_\Phi, \lambda_\Sigma, v, z; u)$, $\lambda_\Delta \subseteq \lambda_\Phi$, $y = v$, $z = u$, we get $Red(\lambda_\Delta, \lambda_\Sigma, y, z; z)$, which together with (*) shows $R(\Delta, y)(\Sigma, z)(\Psi, z)$.

By $Red(\lambda_\Gamma, \lambda_\Delta, x, y; v)$, we get $x \leq y$ and hence $x \leq z = u$. For the case of **E-W**, by hypothesis, we also have that

$\lambda_\Gamma \cap \lambda_\Delta = \emptyset$ and $(\lambda_\Gamma \cup \lambda_\Delta) \cap \lambda_\Sigma = \lambda_\Phi \cap \lambda_\Sigma = \emptyset$,

so that $\lambda_\Gamma \cap \lambda_\Psi = \lambda_\Gamma \cap (\lambda_\Delta \cup \lambda_\Sigma) = \emptyset$. Thus, we can conclude that

$Red(\lambda_\Gamma, \lambda_\Psi, x, z; u)$.

which, together with (**), shows $R(\Gamma, x)(\Psi, z)(\Pi, u)$.This concludes the proof.

Let us proof property **(4)** for **E** and **E-W**. Let (Γ, x) be any world, let us consider the world $(\emptyset, x)$; we have

$\Gamma \cup \emptyset = \Gamma$, $Red(\lambda_\Gamma, \emptyset, x, x; x)$.

This shows that $R(\Gamma, x)(\emptyset, x)(\emptyset, x)$ holds. Moreover, for any (Δ, y), (Σ, z), if $R(\emptyset, x)(\Delta, y)(\Sigma, z)$, holds then,

$\emptyset \cup \Delta = \Sigma$, $Red(\emptyset, \lambda_\Delta, x, y; z)$, which implies $y = z$.

Since $\mathbf{v_0} \leq x$, we also have $Red(\emptyset, \lambda_\Delta, \mathbf{v_0}, y; z)$. We have shown that $R(\emptyset, \mathbf{v_0})(\Delta, y)(\Sigma, z)$, that is $R0(\Delta, y)(\Sigma, z)$. □

Proposition 5.6.12 *For each* $(\Gamma, x) \in W$*, and goal* G*, we have:*

- *(a)* $M, (\Gamma, x) \models G$ *iff* $\Gamma, \lambda_\Gamma \vdash^? x : G$ *succeeds.*
- *(b) If* $\Delta = \{u : D_1, \ldots, u : D_m\}$*, then* $M, (\Delta, u) \models \bigwedge_{i=1}^m D_i$.

Proof. We prove both directions of (a) and (b) by simultaneous induction on the structure of G. Let us consider (a) first. If G is an atom, then the claim holds by definition. The cases of $\wedge$ and $\vee$ are easy and left to the reader. We consider the cases of $\otimes$ and $\rightarrow$, for all systems except for **BCK**. We leave to the reader to modify (easily) the proof to cover the case of **BCK**. Let $G \equiv G_1 \otimes G_2$.

- ($\Rightarrow$)) Suppose $M, (\Gamma, x) \models G_1 \otimes G_2$, then there are (Δ, y) and (Σ, z), such that $R(\Delta, y)(\Sigma, z)(\Gamma, x)$ and

  $M, (\Delta, y) \models G_1$ and $M, (\Sigma, z) \models G_2$.

  By definition of R, we have:

  (*) $\Delta \cup \Sigma = \Gamma$, whence $\lambda_\Gamma = \lambda_\Delta \cup \lambda_\Sigma$ and
  (**) $Red(\lambda_\Delta, \lambda_\Sigma, y, z; x)$.

  By induction hypothesis we get:

  $\Delta, \lambda_\Delta \vdash^? y : G_1$ and $\Sigma, \lambda_\Sigma \vdash^? z : G_2$.

  So that from (*), by monotony we obtain:

  (***) $\Gamma, \lambda_\Delta \vdash^? y : G_1$ and $\Gamma, \lambda_\Sigma \vdash^? z : G_2$.

  By definition, from (*), (**), (***), we have $\Gamma, \lambda_\Gamma \vdash^? z : G$. Since either $x = z$, or $z \in \lambda_\Gamma$ in case of **R**, **L**, by Proposition 5.3.5, we obtain

  $\Gamma, \lambda_\Gamma \vdash^? x : G$.

- ($\Leftarrow$) Let $\Gamma, \lambda_\Gamma \vdash^? x : G_1 \otimes G_2$ succeed; by definition there are γ_1, γ_2, x_1 and x_2, such that $\lambda_\Gamma = \gamma_1 \cup \gamma_2$, and

  (1) $Red(\gamma_1, \gamma_2, x_1, x_2; x)$,
  (2) either $x_2 \in \gamma_2$ (in case **L**, **R**), or $x = x_2$ (in all other cases),
  (3) $\Gamma, \gamma_i \vdash^? x_i : G_i$ succeed, for $i = 1, 2$.

  Let $\Gamma_i = \{x : D \in \Gamma \mid x \in \gamma_i\}$, with $i = 1, 2$, so that $\gamma_i = \lambda_{\Gamma_i}$ and

  (4) $\Gamma = \Gamma_1 \cup \Gamma_2$;

  then by proposition 5.3.1, we have

$\Gamma_i, \gamma_i \vdash^? x_i : G_i$, for $i = 1, 2$.

From (2), we get that, according to each system, either $x \in \gamma_i$ (**R**,**L**), or $max(\gamma_i) = x_i$ (**CL**, **T-W**, **T**) , or $max(\gamma_i) \leq x_i$ (**E**, **E-W**). Thus, $(\Gamma_i, x_i) \in W$ and by induction hypothesis, we have

(5) $M, (\Gamma_1, x_1) \models G_1$ and $M, (\Gamma_2, x_2) \models G_2$.

From (1), (2), (4), we can conclude that $R(\Gamma_1, x_1)(\Gamma_2, x_2)(\Gamma, x)$ holds, whence by (5) we get

$M, (\Gamma, x) \models G_1 \otimes G_2$.

Let us see the case of $\rightarrow$.

- ($\Rightarrow$) Suppose $M, (\Gamma, x) \models (D_1 \wedge \ldots \wedge D_m) \rightarrow G$. Let $\Delta = \{u : D_1, \ldots, u : D_m\}$, with $u > x$, and let $\Sigma = \Gamma \cup \Delta$. For all systems, we have:

  (*) $R(\Gamma, x)(\Delta, u), (\Sigma, u)$.

  By (b), we know that

  (**) $M, (\Delta, u) \models \bigwedge_{i=1}^m D_i$.

  From the hypothesis of ($\Rightarrow$), by (*) and (**), we get that $M, (\Sigma, u) \models G$, whence by induction hypothesis,

  $\Gamma \cup \{u : D_1, \ldots, u : D_m\}, \lambda_\Gamma \cup \{u\} \vdash^? u : G$, succeeds,

  since $\Sigma = \Gamma \cup \{u : D_1, \ldots, u : D_m\}$. Thus, $\Gamma, \lambda_\Gamma \vdash^? x : (D_1 \wedge \ldots \wedge D_m) \rightarrow G$ succeeds.

- ($\Leftarrow$) Let $\Gamma, \lambda_\Gamma \vdash^? x : (D_1 \wedge \ldots \wedge D_m) \rightarrow G$ succeed. Let $u > x$, from the hypothesis, we have

  (1) $\Gamma \cup \{u : D_1, \ldots, u : D_m\}, \lambda_\Gamma \cup \{u\} \vdash^? u : G$ succeeds.

  Now let $R(\Gamma, x)(\Delta, y)(\Sigma, z)$ and $M, (\Delta, y) \models \bigwedge_{i=1}^m D_i$. Thus $M, (\Delta, y) \models D_i$, for $i = 1, \ldots, m$. It easy to see that

  (2) $Comp^S(\Gamma, \lambda_\Gamma, x, u; \Delta, \delta, y)$ holds.

  (3) Let D_i is an atom q_i, then by induction hypothesis we have:

  $\Delta, \lambda_\Delta \vdash^? y : q_i$ succeed.

  (4) Let $D_i = G_i \rightarrow q_i$. By hypothesis we have $M, (\Delta, y) \models G_i \rightarrow q_i$. Let $\Psi, \lambda_\Psi \vdash^? v : G_i$ succeed and $Red^S(\lambda_\Delta, \lambda_\Psi, y, v; v)$. We easily get that

  $R(\Delta, y)(\Psi, v)(\Delta \cup \Psi, v)$ holds.

  By induction hypothesis of (a), we have $M, (\Psi, v) \models G_i$, whence $M, (\Delta \cup \Psi, v) \models q$, so that by the induction hypothesis of (a) again, we can conclude that

  $\Delta \cup \Psi, \lambda_\Delta \cup \lambda_\Psi \vdash^? v : q_i$ succeeds.

  By (1)-(4) all hypotheses of Proposition 5.6.8 are satisfied, thus we obtain that

  (5) $\Gamma \cup \Delta, \lambda_\Gamma \cup \lambda_\Delta \vdash^? y : G$ succeeds.

By hypothesis, we have $\Sigma = \Gamma \cup \Delta$, and whenever it matters $y = z$, thus by the induction hypothesis on (5), we finally obtain

$M, (\Sigma, z) \models G$.

This concludes the proof of (a).

Now we prove (b). Let $\Delta = \{u : D_1, \ldots, u : D_m\}$, then we show that $M, (\Delta, u) \models D_i$ for each D_i. If D_i is an atom, then we have $\Delta, \lambda_\Delta \vdash^? u : D_i$ succeeds, then the claim follows by induction hypothesys of (a). Let D_i be of the form $G_i \to q_i$. We want to show that $M, (\Delta, u) \models G_i \to q_i$. To this purpose, let $R(\Delta, u)(\Phi, v)(\Psi, w)$ and $M, (\Phi, v) \models G_i$. We can apply the induction hypothesis of (a) and get that

$\Phi, \lambda_\Phi \models w : G_i$ succeeds.

By hypothesis, we also have that $u : G_i \to q_i \in \Psi$, moreover we have $Red(\lambda_\Delta, \lambda_\Phi, u, v)$ (and $v = w$, in the cases it matters). Since, $\lambda_\Psi = \lambda_\Delta \cup \lambda_\Phi$, we easily obtain that the conditions for applying reduction wrt. $u : G_i \to q_i$ are matched. Thus we can conclude that $\Psi, \lambda_\Psi \vdash^? w : q_i$ succeeds, whence $M, (\Psi, w) \models q_i$, by the induction hypothesis of (a). This concludes the proof.

□

Theorem 5.6.13 (Completeness) *Let Γ be a labelled set of D-formulas with labels $\lambda_\Gamma = \{x_1, \ldots, x_k\}$ ordered as shown. Let $S_i = \{D \mid x_i : D \in \Gamma\}$ for $i = 1, \ldots, k$, $x = max(\lambda_\Gamma)$, and let G be a G-formula, then we have*

if $\models_S (\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \to G$, then $\Gamma, \lambda_\Gamma \vdash^? x : G$ succeeds.

Proof. By contraposition, suppose that $\Gamma, \lambda_\Gamma \vdash^? x : G$ does not succeed. Then by proposition 5.6.12, we have that

(1) $M, (\Gamma, x) \not\models G$.

Let us consider the following databases $\Gamma_i \subseteq \Gamma$ corresponding to S_i:

$\Gamma_1 = \{x_1 : D \mid x_1 : D \in S_1\}, \ldots \Gamma_k = \{x_k : D \mid D \in S_k\}$,

then we have: $(\Gamma_i, x_i) \in W$, for $i = 1, \ldots, n$. By lemma 5.6.12, we also have that

$M, (\Gamma_i, x_i) \models D$ for each $D \in S_i$ and for $i = 1, \ldots, k$, whence
(*) $M, (\Gamma_i, x_i) \models \bigwedge S_i$, for $i = 1, \ldots, k$.

It is easy to see that $Red^S(\gamma_{\Gamma_1}, \gamma_{\Gamma_2}, x_1, x_2; x_2)$ holds and that $(\Gamma_1 \cup \Gamma_2, x_2) \in W$, so that also

$R(\Gamma_1, x_1)(\Gamma_2, x_2)(\Gamma_1 \cup \Gamma_2, x_2)$ holds.

By (*), we can conclude that

$M, (\Gamma_1 \cup \Gamma_2, x_2) \models \bigwedge S_1 \otimes \bigwedge S_2$.

We can repeat the same argument and show that

$R(\Gamma_1 \cup \Gamma_2, x_2)(\Gamma_3, x_3)(\Gamma_1 \cup \Gamma_2 \cup \Gamma_3, x_3)$ holds,

so that we get $M, (\Gamma_1 \cup \Gamma_2 \cup \Gamma_3, x_3) \models \bigwedge S_1 \otimes \bigwedge S_2 \otimes \bigwedge S_3$ holds. Proceeding in this way, we finally get

(2) $M, (\Gamma, x) \models \bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k$.

Since $R(\emptyset, \mathbf{v_0})(\Gamma, x)(\Gamma, x)$, by (1) and (2), we get that

$M, 0 \not\models (\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \to G$,

which completes the proof. □

5.7 A decision procedure for R

In case of logics without contraction, namely, **CL**, **T-W**, **E-W**, **L**, and **BCK**, the proof procedure we have defined give *decision procedure* for the respective systems. It can be easily seen that they always terminate. This is no longer true for the other systems, **R**, **E**, **T**. The implicational fragment of **R** and **E** is decidable, whereas the same fragment of **T** is not known to be decidable. In this section we modify the previous proof-procedure for **R** presented in the previous section in order to turn it into a decision procedure. The implicational fragment of **R** was shown decidable by Kripke [] in a seminal work going of 1958. This result can be generalized to the whole propositional **R**, without distribution of the lattice connectives. It has been proved by Urquhart [] that **R** with distributive conjunction and disjunction is undecidable.

We first reformulate the procedure for **R** with the following modifications:

1. we insert each formula in the database only once, but we keep track of how many copies of each formula are present; those a database is still a set of labelled formulas, with 1-1 relation beween formulas and labels. On the other hand we keep track in the goal label of the multiplicity of the formulas by using *multisets* of labels.

2. we add a loop-checking mechanism which ensure termination of the deduction procedure. The loop-checking mechanism is the same as the one described in chapter 2 for intutionistic implication (see **??**).

3. when we perform a reduction step, we are allowed to reduce the atomic goal wrt. a formula, say $z : A_1 \to \ldots \to A_n \to q$, even if z is not in the goal label, say α, we cancel one occurrence of z from the goal label, and we do require that the α_i be disjointed. This gives a better control of the splitting of the labels, although it is not strictly necessary to get termination.

By this modification, the set of resources α in a query does not record the available resources anymore, it records what resources we have still to use in a proof.

We hope that the loop-checking method we employ in the case of **R** can be applied to the other systems, and in particular to solve the problem of the decidability of the pure implicational fragment of **T**, which is still open.

We first notice that for **R**, (and for all other systems, except for the cases of **E** and **E-W**), the structure of a query can be simplified: in a regular query

$\Gamma, \alpha \vdash^? x : G$.

the occurrence of x is not necessary, as x can be chosen arbitrarily in α[11]. Thus, we will write a query as

[11]The same simplification can be done in all other systems, but **E** and **E-W**, since either x can be chosen arbitrarily in α, or x is uniquely determined as the maximum label in α, thus in both cases, we do not need to keep track of it.

$\Gamma \vdash^? \alpha : G$.

Definition 5.7.1 A query has the form:

$\Gamma, \vdash^? \alpha : G, H$,

where α is a multiset of labels, H is a set of pairs $\{(\alpha_1, q_1), \dots, (\alpha_n, q_n)\}$, each α_i is a multiset of labels and each q_i is an atomic goal. H is called the history. Deduction rules are as follows:

- (success)
$$\Gamma \vdash^? \alpha : q, H$$
immediately succeeds if $\alpha \subseteq [x]$, and $x : q \in \Gamma$;
- (implication1) from
$$\Delta \vdash^? \alpha : A \to B, H$$
we step to
$$\Delta \cup \{x : A\} \vdash^? \alpha \sqcup [x] : B, \emptyset$$
where $x \notin Lab(\Delta)$ if for no label y, $y : A \in \Delta$.
- (implication2) from
$$\Delta \vdash^? \alpha : A \to B, H$$
we step to
$$\Delta \vdash^? \alpha \sqcup [y] : B, H$$
if for some label y, $y : A \in \Delta$.
- (reduction) from
$$\Delta \vdash^? \alpha : q, H$$
if there is some $y : C \in \Delta$, with $C : A_1 \to \dots \to A_k \to q$, we step, for $i = 1, \dots k$, to
$$\Delta \vdash^? \alpha_i : A_i, H \cup \{(\alpha, q)\},$$
provided the following conditions hold:
  1. there is no $(\beta, q) \in H$, such that $\beta \subseteq\!| \alpha$;
  2. $(\bigsqcup_{i=1}^{k} \alpha_i) = \alpha - [y]$.

■

The loop-checking condition is expressed in condition (1) of reduction rule. The idea of loop-checking is the following: we stop the computation when we perform a reduction step and we re-ask the same atomic goal from a database, which (possibly) contains more copies of some formulas than the database in the query in which the atomic goal was asked the previous time. Intuitively, if the latter query succeeds (with the bigger label), then the former succeeds, because of contraction; thus, there is no reason to carry on such a branch.

We show some examples of loop detection.

Example 5.7.2 Let

$$x : p \to p \vdash^? [x] : p, \emptyset.$$

We step to:

$$x : p \to p \vdash^? \emptyset : p, \{([x], p)\},$$

and then to:

$$x : p \to p \vdash^? \emptyset : p, \{([x], p), (\emptyset, p)\},$$

at this point the computation is stuck by condition (1) on reduction. ■

Example 5.7.3 Let

$$x : (q \to p) \to p \vdash^? [x] : p, \emptyset.$$

We step to:

$$x : (q \to p) \to p \vdash^? \emptyset : q \to p, \{([x], p)\}$$

and then to:

$$x : (q \to p) \to p, y : q \vdash^? [y] : p, \emptyset,$$

then to:

$$x : (q \to p) \to p, y : q \vdash^? [y] : q \to p, \{([y], p)\},$$

and then to:

$$x : (q \to p) \to p, y : q \vdash^? [y, y] : p, \{([y], p)\},$$

now the computation is stuck since $[y] \subseteq| [y, y]$. ■

Example 5.7.4 We show that the following formula is a theorem of **R**:

$$(a \to b \to b) \to a \to a \to b \to b.$$

we start with

$$\emptyset \vdash^? \emptyset : (a \to b \to b) \to a \to a \to b \to b,$$

after a few steps by implication rule, we arrive to

$$\Delta \vdash^? [x_1, x_2^2, x_3] : b,$$

where

$$\Delta = \{x_1 : a \to b \to b, x_2 : a, x_3 : b\},$$

then we go on by reduction wrt. $x_1 : a \to b \to b$. Notice that no other rule is applicable. We generate

$$(1)\ \Delta \vdash^? [x_2] : a, \quad (2)\ \Delta \vdash^? [x_2, x_3] : b$$

query (1) immediately succeeds; we reduce query (2) wrt. $x_1 : a \to b \to b$ again (no other rule is applicable), and we get:

$$(3)\ \Delta \vdash^? [x_2] : a, \quad (4)\ \Delta \vdash^? [x_3] : b.$$

Both queries immediately succeeds.

This formula was considered as an efficiency test for proof systems for Relevant logic (see [Thistlewaite et al. 88])[12]. As we have mentioned in the introduction, the difficulty is that some non-atomic data has to be used twice. In a naive implementation of sequent systems one has to use contraction to duplicate the formula $a \to b \to b$, and this must be done at the beginning of the proof. ■

[12]A proof of the above formula by the theorem prover presented in [Ohlbach and Wrighston 83], based on a semantical translation of relevance logic in classical logic, took about 10 minutes of CPU time!

It is not true that the first time the goal repeats in one branch, we are in a loop, but what we can prove is that: if there is a loop (that is a branch which can be continued forever), then eventually, we will find a (β, q) and a successive (α, q) with $\beta \subseteq\mid \alpha$.

Let us call P1 the procedure without loop-checking, that is the procedure in which the condition 1 in reduction rule is ignored. In P1 the presence of the history is clearly immaterial. Let us call P2 the proof procedure with loop-checking.

Theorem 5.7.5 *The procedure P1 is sound and complete for* **R**.

Proof. Let P_old be the proof system of section 2 (with the obvious simplification introduced in this section). We prove that P1 is equivalent to P_old. It helps to simplify the proof by introducing an intermediate proof system called P_int. P_int works the same as P_old, but the conditions on success and on reduction are changed as follows (the α's are sets in this context):

(Success) $\alpha \subseteq \{x\}$, and $x : q \in \Gamma$;
(Reduction) $(\bigcup_{i=1}^{k} \alpha_i) = \alpha - \{y\}$, $\alpha_i \cap \alpha_j = \emptyset$, and we do not require $y \in \alpha$.

We first prove the equivalence of P_old and P_int.

($\Leftarrow$) We show that:

If $Q = \ \Delta \vdash \alpha : G$ succeeds in P_old, then it succeeds in P_int.

This claim is proved by induction on the height of a successful derivation of Q in P_old. The only non-trivial case is that one of reduction: We need the following fact on P_int, whose proof is by an easy induction on derivations and is left to the reader:

(**fact**) If $\Sigma \ \vdash^? \ \sigma : G$ succeeds and $\delta \subseteq \sigma$, then also $\Sigma \ \vdash^? \ \delta : G$ succeeds.

Now let a successful derivation of Q proceed by reduction, then G is an atom q, there is some $y : C \in \Delta$, with $C : \ A_1 \to \ldots \to A_k \to q$, there are α_i, and x_i for $i = 0, \ldots, k$ such that:

1. $\alpha_0 = \{y\}$,
2. $\bigcup_{i=0}^{k} \alpha_i = \alpha$.

and for $i = 1, \ldots k$, we step to

$Q_i = \ \Delta, \vdash^? \ \alpha_i : A_i$.

By induction hypothesis, Q_i succeed in P_int. Now let

$\alpha'_i = \alpha_i - \bigcup_{0 \leq j < i} \alpha_j$, for $i = 1, \ldots, k$.

We have that $\alpha'_i \subseteq \alpha_i$, $\alpha_i \cap \alpha_j = \emptyset$ and $(\bigcup_{i=1}^{k} \alpha_i) = \alpha - \{y\}$. Moreover, by the (**fact**) above, each

$Q'_i = \ \Delta, \alpha'_i, \vdash^? \ x_i : A_i$ succeeds.

Hence, according to procedure P_int from Q we may step to Q'_i and succeed.

($\Leftarrow$) We show that:

if $Q = \ \Delta \ \vdash^? \ \alpha : G$ succeeds in P_int, then there is a set β, such that $Q' = \ \alpha \subseteq \beta \subseteq Lab(\Delta)$ such that $\Delta \ \vdash^? \ \beta : G$ succeeds in P_old. In particular, if $\alpha = Lab(\Delta)$ and $\Gamma \ \vdash^? \ \alpha : A$ succeeds in P_int, then it succeeds also in P_old.

Again the proof is by induction on derivation, we only show in some detail the case of reduction. Let a successful derivation of Q proceed by reduction, then G is an atom q, there is some $y : C \in \Delta$, with $C : A_1 \to \ldots \to A_k \to q$, there are α_i, for $i = 0, \ldots, k$ such that $(\bigcup_{i=1}^{k} \alpha_i) = \alpha - \{y\}$ and $\alpha_i \cap \alpha_j = \emptyset$, for all i, j, and for $i = 1, \ldots k$, we step to

$$Q_i = \Delta \vdash^? \alpha_i : A_i.$$

By induction hypothesis, there are some β_i, with $\alpha_i \subseteq \beta_i \subseteq Lab(\Delta)$, such that

$$Q_i' = \Delta \vdash^? \beta_i : A_i$$

succeed in P_old, we take $\beta = \bigcup_i \beta_i \cup \{y\}$, and from Q' we step to Q_i' and succeed.

Now we show the equivalence between P_int and P1, we omit the history, since it does not play any role. Let Δ be a database for P_int, define

$x \equiv_\Delta y$ iff for some formula A, $x : A \in \Delta$ and $y : A \in \Delta$.

Given a set of labels α, let $m_\Delta(\alpha)$ be the multiset, whose support is $\alpha_{/\equiv_\Delta}$ (the quotient of α wrt. $\equiv_\Delta$) and such that, for all x, $m_\Delta(\alpha)(x) =$ the cardinality of $[x]_{\equiv_\Delta}$ in α. We also let $m(\Delta)$ be the set of formulas of Δ relabelled by labels of $Lab(\Delta)_{/\equiv_\Delta}$. Since there is a 1-1 mapping between formulas of $m(\Delta)$ and labels, we can subscribe labels with the formulas they label. For example, let $\Delta = \{x : A, y : A, z : B, u : C\}$ and $\alpha = \{x, y, z\}$, then $m_\Delta(\alpha) = [x_A^2, x_B]$, and $m(\Delta) = \{x_A : A, x_B : B, x_C : C\}$.

($\Rightarrow$) It is an easy exercise to show that:

If $\Gamma \vdash^? \alpha : G$ succeeds in P_int, then $m(\Gamma) \vdash^? m_\Delta(\alpha) : G$ succeeds in P1.

The proof proceedes by induction on the height of a derivation of the first query in P_int. In the case of an implication goal $A \to B$ we distinguish the two cases when a copy of A is already present in Δ from that one in which it is not, and we apply rules implication1 or implication2 accordingly in the corresponding derivation in P1. In case of reduction, we can esily conclude from the facts that, since for all i, j $\alpha_i \cap \alpha_j = \emptyset$, we have

$m_\Delta(\bigcup_i \alpha_i) = \bigsqcup_i (m_\Delta(\alpha_i))$, and also
$m_\Delta(\alpha - \{y\}) = m_\Delta(\alpha) - m_\Delta(y)$.

($\Leftarrow$) Conversely, let $Q = \Gamma \vdash^? \alpha : G$ be a query in P1, we turn α into a set $s(\alpha)$ by renaming each copy of a label in α by a distinct label, e.g. let $\alpha = [x^2, y^3, z]$, then we may take $s(\alpha) = \{x_1, x_2, y_1, y_2, y_3, z\}$. We call s a set- mapping. Then, given s and α, we expand Δ to a new database $s_\alpha(\Delta)$, by inserting $\alpha(x)$ copies of $x : A \in \Delta$, renamed with the labels drawn from $s(\alpha)$. For instance, let α as above and $\Delta = \{x : A, y : B, z : C, u : D\}$, then

$$s_\alpha(\Delta) = \{x_1 : A, x_2 : A, y_1 : B, y_2 : B, y_3 : B, z : C, u : D\}.$$

The way we rename the elements of the multiset is arbitrary, what matters is that the cardinality of the resulting set is the same as the cardinality of the multiset (counting repetitions). Again it is an easy exercise to show that:

If $\Gamma \vdash^? \alpha : G$ succeeds in P1, then there is a set-mapping s, such that $s_\alpha(\Gamma) \vdash^? s(\alpha) : G$ succeeds in P_int.

As before, the proof proceedes by induction on the height of a derivation of the first query in P1. We only give some details of the case of reduction. Let $Q = \ \Gamma \ \vdash^? \ \alpha : q$ be derived by reduction, then for some $y : C \in \Delta$, with $C : \ A_1 \rightarrow \ldots \rightarrow A_k \rightarrow q$, there are multiset α_i, for $i = 0, \ldots, k$ such that $(\bigsqcup_{i=1}^{k} \alpha_i) = \alpha - [y]$ and for $i = 1, \ldots k$, we step to

$Q_i = \ \Delta \ \vdash^? \ \alpha_i : A_i$.

By induction hypothesis there are set-mappings s^i, such that

$s^i_{\alpha_i}(\Gamma) \ \vdash^? \ s^i(\alpha_i) : A_i$ succeeds in P_int.

We can find a set mapping s such that:

- $s(\alpha_i) \cap s(\alpha_j) = \emptyset$ and
- $\bigcup_i s(\alpha_i) = s(\alpha) - s([y])$.

Since renaming labels does not matter and $s(\alpha_i)$ and $s_i(\alpha_i)$ are just renaming of the same variables, we get that:

$s_{\alpha_i}(\Gamma) \ \vdash^? \ s(\alpha_i) : A_i$ succeeds in P_int,

but then, by monotony, also

$Q'_i = s_\alpha(\Gamma) \ \vdash^? \ s(\alpha_i) : A_i$ succeeds in P_int,

succeeds in P_int. Hence from $s_\alpha(\Gamma) \ \vdash^? \ s(\alpha) : q$ we can step to Q'_i and succeed.

□

We are going to show that the proof procedure with loop- checking is sound and complete for **R**. Soundness is obvious, given the soundness of the more liberal proof procedure without loop-checking (the previous theorem). We need a few lemmas, whose proofs are easy and hence omitted.

Lemma 5.7.6 *If* $Q = \ \Gamma \ \vdash^? \ \alpha : G, H$ *succeeds in P1 by a derivation* $\mathcal{D}$*, and* $H' \subseteq H$*, then also* $\Gamma \ \vdash^? \ \alpha : G, H'$ *by an isomorphic derivation.*

Definition 5.7.7 Given two derivations $\mathcal{D}$ and $\mathcal{D}'$, we say that $\mathcal{D}$ is *embeddable* in $\mathcal{D}'$ if

1. for each query $Q = \Gamma \ \vdash^? \ \gamma : G, H$ in $\mathcal{D}'$, there is a corresponding query $e(Q) = \Gamma \ \vdash^? \ \gamma' : G, H'$ in $\mathcal{D}'$, such that $\gamma \subseteq \gamma'$, and for each $(\delta, r) \in H$ there is $(\delta', r) \in H'$, with $\delta \subseteq \delta'$.

2. if Q' is a child of Q in $\mathcal{D}$, then $e(Q)$ is a child of $e(Q')$ in $\mathcal{D}'$.

■

In other words, $\mathcal{D}$ is isomorphic to a subtree of $\mathcal{D}'$, and the label and the histories of the corresponding queries are related as shown in item (1).

Lemma 5.7.8 *If* $Q = \ \Gamma \ \vdash^? \ \alpha : G, H$ *succeeds in P1 by a derivation* $\mathcal{D}$ *and* $\beta \subseteq \alpha$*, then also* $Q' = \Gamma \ \vdash^? \ \beta : G, H$ *succeeds in P1 by a derivation* $\mathcal{D}'$ *embeddable in* $\mathcal{D}$*.*

Given a derivation $\mathcal{D}$ in P1, we say that a query Q in $\mathcal{D}$ is a *violation* if reduction rule is applied to Q without respecting the loop-checking condition (1) in reduction rule. Let $v(\mathcal{D})$ be the number of violations in $\mathcal{D}$.

Theorem 5.7.9 *The procedure with loop-checking is sound and complete for* **R**.

Proof. We prove completeness. We show how to turn a successful P1-derivation $\mathcal{D}$ of a query Q:

$$\Sigma \vdash^? \chi : G, \emptyset.$$

into a successful P2-derivation $\mathcal{D}^*$ of Q, that is, into a successful P1-derivation $\mathcal{D}^*$ such that $v(\mathcal{D}^*) = 0$. Let us define the *rank* of a derivation $\mathcal{D}$, denoted by $r(\mathcal{D})$, as the pair $(h(\mathcal{D}), v(\mathcal{D}))$. We consider ranks lexicographically ordered, and we prove, by induction on $r(\mathcal{D})$, that if Q succeeds by a derivation $\mathcal{D}$ of rank (h, v), then it succeeds by a derivation $\mathcal{D}^*$ embeddable in $\mathcal{D}$, of rank $r(\mathcal{D}') = (h', 0)$, (with $h' \leq h$).

The case of $r(\mathcal{D}) = (h, 0)$ is trivial and the case $r(\mathcal{D}) = (0, v)$ is impossible. Thus, we are left with the case $r(\mathcal{D}) = (h, v)$, with $h, v > 0$.

Inspecting $\mathcal{D}$ from the root downward, choose a query Q' which violates the loop-checking constraint. Let

$$Q' = \Gamma \vdash^? \alpha : q, H'.$$

If Q' is a violation, then there is $(\beta, q) \in H'$, such that $\beta \subseteq\mid \alpha$. This means that above Q', on the path from Q to Q' in $\mathcal{D}$, there occurs a query:

$$Q'' = \Delta \vdash^? \beta : q, H''.$$

Let $\mathcal{D}_1$ be the subtree rooted in Q' and $\mathcal{D}_2$ be the subtree rooted in Q''; let $r(\mathcal{D}_1) = (h_1, v_1)$ and $r(\mathcal{D}_2) = (h_2, v_2)$. We have $h_1 < h_2 \leq h$, moreover $v_1 < v_2$, whence:

$$r(\mathcal{D}_1) < r(\mathcal{D}_2) \leq r(\mathcal{D}).$$

Since $(\beta, q) \in H'$, the history has not been cleared (because of the implication1 rule) along the path from Q'' to Q'; but this entails that no new formulas have been introduced in Δ, that is $\Gamma = \Delta$ and $H'' \subseteq H'$. By the previous lemmas, we have that the query

$$Q^* = \Gamma \vdash^? \beta : q, \emptyset$$

succeeds by a derivation $\mathcal{D}_1'$ embeddable in $\mathcal{D}_1$ of height $h_1' \leq h_1 < h$, whence

$$r(\mathcal{D}_1') < r(\mathcal{D}).$$

We may apply the induction hypothesis on Q^* and obtain that Q^* succeeds by a derivation $\mathcal{D}_1^*$ of rank $(h_1^*, 0)$, with $h_1^* \leq h_1' \leq h_1 < h_2$; moreover, $\mathcal{D}_1^*$ is embeddable in $\mathcal{D}_1'$, and hence in $\mathcal{D}_1$. Now let $\mathcal{D}_2^*$ be obtained from $\mathcal{D}_1^*$ by inserting H'' in Q^*, that is the top query of $\mathcal{D}_2^*$ is Q'', and the history is propagated accordingly. We have that also $\mathcal{D}_2^*$ is embeddable in $\mathcal{D}_1$, since $H'' \subseteq H'$. Obviously, derivation $\mathcal{D}_2^*$ is successful.

We have that $r(\mathcal{D}_2^*) = (h_1^*, v_2^*)$, for some v_2^*. It is sufficient to show that it must be $v_2^* < v_2$. By hypothesis, we have that $v_1 < v_2$. We show that $v_2^* \leq v_1$. To this aim, let Q_t be a violation in $\mathcal{D}_2^*$. Since $\mathcal{D}_1^*$ does not contain violations, Q_t must be a violation because of the history H'' which has been inserted in Q'' and propagated in accordance with the deduction rules. It must be that Q_t has the following form:

$$Q_t = \Gamma \vdash^? \delta : r, H_t,$$

where $H'' \subseteq H_t$ and there is $(\gamma, r) \in H''$, with $\gamma \subseteq\mid \delta$. Notice that the database must be Γ, otherwise the history would have been cleared. Since $\mathcal{D}_2^*$ is embeddable in $\mathcal{D}_1$, there is a corresponding query Q_t' in $\mathcal{D}_1$ of the form:

$Q'_t = \Gamma \vdash^? \delta' : r, H'_t$,

with $H'' \subseteq H'_t$ and $\delta \subseteq \delta'$. We show that Q'_t must be a violation in $\mathcal{D}_1$, that is $\gamma \subseteq\mid \delta'$. Suppose it is not the case. First observe that since $(\gamma, r) \in H'_t$, it must be

(*) $\bar{\delta'} \subseteq \bar{\gamma}$,

this follows from the fact that, if new labels had been introduced in δ', the history would have been cleared, and this is not the case. Suppose that $\gamma \not\subseteq\mid \delta'$; by (*), we then have

either $\bar{\delta'} \subset \bar{\gamma}$ or $\delta' \subset\mid \gamma$.

In the former case we have $\bar{(\delta')} \subset \bar{\gamma} = \bar{\delta}$, against the fact that $\delta \subseteq \delta'$. In the latter case, we have $\delta' \subset\mid \gamma \subseteq\mid \delta$, and hence $\delta' \subset\mid \delta$, contradicting again the fact that $\delta \subseteq \delta'$. We have shown that, for each violation in $\mathcal{D}_2^*$ there is a corresponding violation in $\mathcal{D}_1$, hence $v_2^* \leq v_1$.

We obtain a new successful derivation $\mathcal{D}^*$ of the original query Q, by replacing $\mathcal{D}_2$ by $\mathcal{D}_2^*$ in $\mathcal{D}$. Since $h_1^* < h_2$ and $v_2^* < v_2$, we have that $h(\mathcal{D}^*) \leq h$ and $v(\mathcal{D}^*) < v$, whence $r(\mathcal{D}^*) < r(\mathcal{D})$. To conclude the proof, we apply the induction hypothesis to $\mathcal{D}^*$.

$\square$

Remark 5.7.10 Notice that, the transformation never increases the size of a derivation. This means, as expected, that loop-free derivations are usually shorter than derivation in which violations are allowed.

We are going to prove that the procedure with loop- checking always terminates. To this purpose we need a proeperty of sequences of multisets (Kripke's lemma). Let $\alpha_1, \alpha_2, \ldots, \alpha_i, \alpha_{i+1} \ldots$ be a sequence of multisets on a finite set S, we say that the sequence is *irredundant* if, whenever $i < j$, it is not the case that $\alpha_i \subseteq\mid \alpha_j$.

Lemma 5.7.11 (Kripke) *Any irredundant sequence of multisets $\alpha_1, \alpha_2, \ldots, \alpha_i, \alpha_{i+1} \ldots$ on a finite set is finite.*

Theorem 5.7.12 (Termination) *Procedure P2 always terminates.*

Proof. Let $Q = \Delta \vdash^? \alpha : G, \emptyset$ be a query, we show that any derivation tree $\mathcal{D}$ with root Q in P2 is finite. We argue by absurdity. Suppose $\mathcal{D}$ is infinite. Since $\mathcal{D}$ is a finitely-branching tree, (by König's lemma), it has an infinite branch $\mathcal{B}$. Let $Q' = \Sigma \vdash^? \gamma : G', H$ be any query in $\mathcal{B}$. Since the number of subformulas of Δ and G is finite, say k, no more than k distinct labels, say $S = \{x_1, \ldots, x_k\}$, can occur in γ. That is to say, γ is a finite multiset on S. Moreover, we can assume also that $Lab(\Delta) \subseteq Lab(\Sigma) \subseteq S$.

There must be a query $Q_i = \Gamma_i \vdash^? \alpha_i : G_i, H_i$ on $\mathcal{B}$, such that

For all $Q_j = \Gamma_j \vdash^? \gamma_j : G_j, H_j$ on $\mathcal{B}$ with $j \geq i$, $\Gamma_j = \Gamma_i$.

This follows from the fact that, for every Γ_j, $Lab(\Gamma_j) \subseteq S$, and from the fact that

if $\Gamma \vdash^? \alpha : G, H$ precedes $\Gamma' \vdash^? \alpha' : G', H'$, then $\Gamma \subseteq \Gamma'$.

But this implies that, from Q_i onwards on $\mathcal{B}$, the history will never be cleared, that is:

(*) for all $j \geq i$, $H_j \subseteq H_{j+1}$.

Since $\mathcal{B}$ is infinite, after Q_i, there must be infinitely many queries Q'_j with the same atomic goal q'. Let us consider the infinite subsequence of such queries $Q'_j = \Gamma \vdash^? \gamma'_j : q', H'_j$. Now we are ready to derive a contradiction. By Kripke's lemma, the infinite sequence of multisets (on S) $\gamma'_i, \gamma'_{i+1}, \ldots$ must contain some γ'_l and γ'_k such that $l < k$ and $\gamma'_l \subseteq\!| \gamma'_k$; they come from two queries:

$$Q'_l = \Gamma \vdash^? \gamma'_l : q', H_l \quad \text{and} \quad Q'_k = \Gamma \vdash^? \gamma'_k : q', H_k.$$

The query $Q''_{l+1} = \Gamma \vdash^? \gamma'_{l+1} : G_{l+1}, H''_{l+1}$ which follows Q'_l in $\mathcal{B}$ must be obtained from Q'_l by reduction (it may be $Q''_{l+1} = Q'_k$), hence by (*) we have:

$$H''_{l+1} = H'_l \cup \{(\gamma'_l, q')\} \subseteq H'_k.$$

But this implies that $(\gamma'_j, q') \in H'_k$, and hence, by the loop-checking condition, branch $\mathcal{B}$ must end with Q'_k. □

Although the procedure for implicational **R**terminates, its complexity is still under investigation. Urquhart [] has proved an exponential-space lower bound and un upper bound which is primitive-recursive in the Ackermann function for the pure implicational fragment. The latter bound is essentially also a lower bound for the fragment with $\rightarrow, \wedge$. ˇThere is therefore a huge gap between the lower bound and the upper bound for the implicational fragment.

SHALL WE PUT R-MINGLE???? THE FOLLOWING SECTION SHOULD BE DEEPLY REVISED OR OMITTED

5.8 A further case study: the system RM0

The system RM0 is an extension of R which formalize a more liberal discipline on use of formulas. In **R**we have that $\Gamma \vdash A$ if there is a derivation in which every hypothesis in Γ is used. In **RM0**we demand less: for every hypothesis in Γ there must be a derivation of A which uses it. It may happen that no derivation is capable of exhausting all formulas of the database, but if we take together alternative derivations they jointly exhaust all formulas. Let us consider the following example:

$$(a \rightarrow c) \rightarrow (b \rightarrow c) \rightarrow a \rightarrow b \rightarrow c.$$

Suppose we try a derivation of this formula, using the procedure for **R**, we have:

$$\emptyset \vdash^? \emptyset : (a \rightarrow c) \rightarrow (b \rightarrow c) \rightarrow a \rightarrow b \rightarrow c.$$

$$\downarrow$$

$$\vdots$$

$$(1)\ x : a \rightarrow c, y : b \rightarrow c, z : a, u : b \vdash^? \{x, y, z, u\} : c$$

we can use the first formula and step to:

$$(2)\ x : a \rightarrow c, y : b \rightarrow c, z : a, u : b \vdash^? \{y, z, u\} : a$$

and we fail according to the procedure for R. Alternatively at step (1) we can choose $y : b \rightarrow c$ and step to

$$x : a \rightarrow c, y : b \rightarrow c, z : a, u : b \vdash^? \{x, z, u\} : b$$

and we fail again. The above formula is not a theorem of R. According to RM0, we can say "ok the goal a in (2) succeeds (consuming the label z), provided we can consume the remaining labels to prove the original goal, or more generally, any previous goal involved in the derivation. Thus after (2) we can restart by asking:

$$x : a \to c, y : b \to c, z : a, u : b \vdash^? \{y, u\} : c,$$

and then step to

$$x : a \to c, y : b \to c, z : a, u : b \vdash^? \{u\} : b,$$

which succeeds. The principle of RM0 can be informally stated as follows: if the current goal succeeds, but leaves some resources unused, then restart the computation from any previous goal and the corresponding database, trying to consume the unused data.

We first give a procedure which extends that one for **R** by allowing to restart the computation in the above sense. Then, we present an equivalent procedure which use an explicit rule to partition the available resources, rather than the restart rule. This rule which partitions the resources is the Mingle rule, from which the system takes its name.

A remark on terminology, **RM0** is the system obtained by adding to the implicational fragment of R the axiom:

$$A \to A \to A,$$

or equivalently, the axiom

$$(A \to C) \to (B \to C) \to A \to B \to C.$$

If we add these axioms to the whole system **R** we obtain a system called R-Mingle (RM), whose implicational fragment is strictly stronger than **RM0**, and will not be treated here.

WE HAVE TO CHECK THIS

Database and labels are defined as in the previous sections. A query has the form:

$$\Gamma \vdash^? \alpha : A, H,$$

where Γ, and α are as in the case of R, and H, called the history, is a list of triplets of the form

$$H = (\Delta_1, \beta_1, q_1), \ldots, (\Delta_n, \beta_n, q_n),$$

where Δ_i are databases, β_i are labels, and q_i are atoms. We use the append notation and write:

$$H = (\Delta_1, \beta_1, q_1) * \ldots * (\Delta_n, \beta_n, q_n),$$

We adopt a similar policy to the one of section 5.7, i.e. when we perform a reduction step, we are allowed to reduce the atomic goal wrt. a formula, say $z : A_1 \to \ldots \to A_n \to q$, even if z is not in the goal label, say α, then we cancel z from the goal label and we do require that the α_i be disjointed. This makes more efficient the control of the labels. We limit our duscussion to the implicational fragment.

The rules are as follows:

- (success)

$$\Delta \vdash^? \alpha : q, H,$$

  succeeds, if for some x, $x : q \in \Delta$ and $\alpha = x$ or $\alpha = \emptyset$;

- (implication) from
$$\Delta \vdash^? \alpha : C \to G, H$$
we step to
$$\Delta \cup \{x : C\} \vdash^? \alpha \cup \{x\} : G, H'$$
where x is a new label, that is not occurring neither in Δ, nor in α, nor in H, and $H' = H * (\Delta, \alpha \cup \{x\}, G)$ if G is an atom, and $H' = H$ otherwise.

- (reduction) from
$$\Delta \vdash^? \alpha : q, H$$
if there is some $y : C \in \Delta$, with
$$C : A_1 \to \ldots \to A_k \to q,$$
then we step, for $i = 1, \ldots k$, to
$$\Delta \vdash^? \alpha_i : A_i, H_i,$$
where

  1. $\alpha_i \cap \alpha_j = \emptyset$, for $i \neq j$;
  2. $\bigcup_{i=1}^{k} \alpha_i = \alpha - \{y\}$.
  3. $H_i = H * (\Delta, \alpha_i, A_i)$ if A_i is an atom, and $H_i = H$ otherwise.

  (Notice that we do not require that $y \in \alpha$).

- (Restart) from from
$$\Delta \vdash^? \alpha : q, H$$
if $H = H_1 * (\Gamma, \beta, r) * H_2 * (\Delta, \alpha : q)$, and the following conditions hold:

  1. for some x, $x : q \in \Delta$,
  2. letting $\alpha' = \alpha - \{x\}$, we have $\alpha' \subseteq \beta$

  then we step to
$$\Gamma \vdash^? \alpha' : r, H'$$
where $H' = H_1 * (\Gamma, \alpha', r)$.

The first condition ensures that q succeeds if we ignore the label. The second condition may be explained by saying that if we select (Γ, β, r) in the history, we commit to "consume" β. That is why the current label α' must be included in β, otherwise we would re-try the goal r with extraneous resources not occurring when it was asked before (represented by β).

Example 5.8.1 We reconsider the previous example:

$$\emptyset \vdash^? \emptyset : (a \to c) \to (b \to c) \to a \to b \to c, \emptyset.$$

$$\downarrow$$

$$\vdots$$

$$\Delta \vdash^? \{x, y, z, u\} : c, (\Delta, \{x, y, z, u\}, c)$$

where $\Delta = \{x : a \to c, y : b \to c, z : a, u : b\}$ we reduce wrt. $x : a \to c$:

$$\Delta \vdash^? \{y, z, u\} : a,\ (\Delta, \{x, y, z, u\}, c) * (\Delta, \{y, z, u\}, a)$$

since $z : a \in \Delta$ and $\{y, u\} \subseteq \{x, y, z, u\}$, we can restart by asking:

$$\Delta \vdash^? \{y, u\} : c,\ (\Delta, \{y, u\}, c)$$

and then step to

$$\Delta \vdash^? \{u\} : b,\ (\Delta, \{y, u\}, c) * (\Delta, \{u\}, b)$$

which immediately succeeds. ■

Another simpler example is

$$\emptyset \vdash^? \emptyset : a \to a \to a, \emptyset$$

$$x : a \vdash^? x : a \to a, \emptyset$$

$$x : a, y : a \vdash^? \{x, y\} : a,\ (\{x : a, y : a\}, \{x, y\}, a)$$

we can apply restart and step to

$$x : a, y : a \vdash^? \{y\} : a,\ (\{x : a, y : a\}, \{y\}, a)$$

which immediately succeeds.

As we have seen in the example above, the query to which we apply the restart rule may be identical to the query from which we restarted, that is if $x : q \in \Gamma$, then from

$$\Gamma \vdash^? \alpha : q, H * (\Gamma, \alpha, q),$$

we can step to

$$\Gamma \vdash^? \alpha - \{x\} : q, H * (\Gamma, \alpha - \{x\}, q).$$

We call this limit type of restart *self-restart*. We will see at the end of this section that we can eliminate this type of restart by changing the label discipline.

From the viewpoint of Gentzen's systems, **RM0** can be obtained by adding to the Gentzen system for the implicational fragment of R the following rule:

$$\frac{\Gamma \vdash A \quad \Delta \vdash A}{\Gamma, \Delta \vdash A} .$$

This rule can be easily reformulated in our context as follows:

(Mingle) From

$$\Delta \vdash^? \alpha : A, H$$

step to

$$\Delta \vdash^? \alpha_1 : A, H_1 \quad \text{and} \quad \Delta \vdash^? \alpha_2 : A, H_2$$

where $\alpha = \alpha_1 \cup \alpha_2$, $\alpha_1 \cap \alpha_2 = \emptyset$, and $H_i = H * (\Delta, \alpha_i, A)$, if A is an atom, and $H_i = H$ otherwise.

That is we can split the label α in two labels and carry on two computations one consuming α_1 and the other consuming α_2. Our restart rule can be seen as a linearization of the Mingle-rule. As we prove below, the Mingle rule is equivalent to the restart rule. To this aim we let P1 be the procedure which use mingle rule and does not use restart, and let P2 be the procedure which use restart and does not use mingle. Technically, we need to consider also a (redundant) derivation procedure P1 + P2 in which both Mingle and restart are allowed. We show how to gradually replace all applications of one of the two rules by the other one, in any given successful derivation. We thereby obtain that that either one or the other rule can be avoided, so that a P1-derivation can be converted into a P2-equivalent one and viceversa.

Proposition 5.8.2 *[Monotonicity] If* $\Gamma \subseteq \Delta$, *then* $\Gamma \vdash^? \alpha : G$ *succeeds with height* h *implies* $\Delta \vdash^? \alpha : G$ *succeeds with height* $h' \leq h$.

Proposition 5.8.3 *Let* α *and* β *be two labels such that* $\beta \subseteq \alpha$, *then for any formula* Γ, G, *and* H $\Gamma \vdash^? \alpha : G, H,$ *with height* h *implies* $\Gamma \vdash^? \beta : G, H$ *with height* $h' \leq h$.

Proposition 5.8.4 *If* $\Gamma \vdash^? \alpha : A, H$ *succeeds in P1, then it has a successful derivation in which Mingle rule is restricted to atoms.*

Theorem 5.8.5 *Let* $\mathcal{D}$ *be a successful derivation in P1(i.e. which uses Mingle) of the query* $K = \Gamma \vdash^? \psi : A, H$, *then there is a successful P2-derivation* $\mathcal{D}'$ *of* K.

Proof. Let us consider a successful derivation $\mathcal{D}$ in P1 of K. We show how to replace every application of the mingle rule by an application of the restart rule. By proposition 5.8.4, we can assume that every application of mingle is restricted to atomic queries. At intermediate steps the derivation built so far will contain application of both restart and mingle (formally, a derivation in P1 + P2). But the intermediate derivation will satisfy the property that no application of restart precedes any application of mingle. More precisely, if restart is applied to N using a previous N', and mingle is applied to N'', then N'' cannot be equal to N', or be a descendant of N'. The initial derivation $\mathcal{D}$ trivially satisfy this property. We stepwise transform the initial derivation by replacing any application of mingle by an application of restart in such a way that the above property is preserved.

We say that an application of mingle to a query Q in $\mathcal{D}$ is *maximal* if there are no applications of mingle to any descendant of Q. Even if the transformed derivation is longer, every transformation step decreases the number of maximal applications of mingle, without introducing new applications of the rule. Thus, the process terminates in a finite number of steps.

We describe a generic transformation step. Suppose $\mathcal{D}$ has the form shown in Fig.5.2.

In the shown derivation, $\alpha = \alpha_1 \cup \alpha_2$, and the subtree T_1 and T_2 do not contain any application of the mingle rule. Moreover, if restart is applied in T_i, then the query used by restart must be a descendant of

$$N_0 : \Gamma \vdash^? \alpha : q, H * (\Gamma, \alpha, q).$$

We let

$$N = \Gamma \vdash^? \alpha_1 : q, H * (\Gamma, \alpha_1, q).$$

T_1 must have a leaf of the form

$$N_i = \Delta \vdash^? \psi : r, H_1^*,$$

$$\begin{tikzcd}
& K \arrow[dl] \arrow[d] \arrow[dr] & \\
\dots & \vdots & \dots \\
& \Gamma \vdash^{?} \alpha : q, H * (\Gamma, \alpha, q) \arrow[dl] \arrow[dr] & \\
\Gamma \vdash^{?} \alpha_1 : q, H * (\Gamma, \alpha_1, q) & & \Gamma \vdash^{?} \alpha_2 : q, H * (\Gamma, \alpha_2, q) \\
T_1 & & T_2
\end{tikzcd}$$

Figure 5.2:

such that for some x, $x : r \in \Delta$ and $\psi = \emptyset$ or $\psi = x$. Notice that in the latter case $x \notin \alpha_2$. Moreover we have

$$H_1^* = H * (\Gamma, \beta, q) * H_1' * (\Delta, \psi, r),$$

where $\beta = \alpha_1$ or $\beta = \alpha_1 - \{z\}$, since the successor of N in T_1 might be obtained by self- restart.

Let T_1' be obtained from T_1 as follows: first add α_2 to the goal-label of every query Q occurring in the path leading from N (included) to N_i, and to the label occurring in triplet corresponding to Q in the history. We have that the root of T_1' is

$$N_0 : \ \Gamma \vdash^{?} \alpha : q, H * (\Gamma, \alpha, q),$$

and the leaf corresponding to N_i is

$$N_i' = \ \Delta \vdash^{?} \alpha_2 \cup \psi : r, H_1,$$

with

$$H_1 = H * (\Gamma, \beta \cup \alpha_2, q) * H_1'' * (\Delta, \alpha_2 \cup \psi, r),$$

where H_1'' is obtained by adding α_2 to the label of each triplet in H_1'. Since $x : r \in \Delta$ and $\gamma = (\alpha_2 \cup \psi) - \{x\} \subseteq \alpha \cup \beta$, we can go on by restart from N_i' by using N_0 or its immediate successor (in the case it is obtained by self-restart) and step to

$$Q = \ \Gamma \vdash^{?} \gamma : q, H * (\Gamma, \gamma, q).$$

We have that either $\gamma = \alpha_2$ or $\gamma = \alpha_2 - \{x\}$. To query Q we append the tree T_2', which is T_2, if $\gamma = \alpha_2$, or it is obtained by deleting the atomic label x from goal labels (and history) of queries of T_2. Here below (Fig.5.3) the resulting derivation is displayed. In this way we have obtained a new derivation $\mathcal{D}'$, which has one (maximal) application of mingle less than the original $\mathcal{D}$. The only difference between the two derivations is below N_0. But the only difference between T_1 and T_1' is that the first one has a branch ending by N_i, whereas the latter has a branch going on with N_i' and then T_2'. If $\mathcal{D}$ is successful, then T_1 and T_2 are successful subtrees. By proposition 5.8.3, T_2' is successful, and hence so is T_1'; we can conclude that $\mathcal{D}'$ is successful. □

Theorem 5.8.6 *Let $\mathcal{D}$ be a successful derivation in P2 (i.e. which uses Restart) of the query $K = \Gamma \vdash^{?} \psi : A, H$, then there is a successful P1-derivation $\mathcal{D}'$ of K (which uses only Mingle).*

$$K$$

$$\swarrow\downarrow\searrow$$

$$\ldots \quad \vdots \quad \ldots$$

$$\Gamma \vdash^{?} \alpha : q, H * (\Gamma, \alpha, q).$$

$$(\ \Gamma \vdash^{?} \alpha_2 \cup \beta : q, H * (\Gamma, \alpha_2 \cup \beta, q))\ )$$

(in case of self restart)

$$T_1'$$

$$\Delta \vdash^{?} \alpha_2 \cup \psi : r, H_1$$

$$\Gamma \vdash^{?} \gamma : q, H * (\Gamma, \gamma, q)$$

$$T_2'$$

Figure 5.3:

Proof. Let us consider a successful derivation $\mathcal{D}$ in P2 of K. We show how to replace every application of restart rule by an application of mingle rule. At intermediate steps the derivation built so far will contain application of both restart and mingle (formally, a derivation in P1 + P2). As before, the intermediate derivation will satisfy the restriction that no application of restart precedes any application of mingle. An application of the restart rule is called *minimal* if it is done to a query N by using a previous query N', and no application of restart to any query on any branch going through N' uses a query which precedes N'. The application is minimal wrt N' in the sense no other else application of restart uses a query "older" than N'. The transformation described below replace a minimal application of restart by an application of mingle, and it yields a derivation of no greater height, in which there is at least one minimal application of restart less than the original one, and no new application of restart. If the starting derivation is successful then it is finite, and by iterating the transformation we replace every application of restart by mingle.

We describe a generic transformation step. Suppose $\mathcal{D}$ has the form shown in Fig.5.4.

In the shown derivation, $\beta' = \beta - \{x\} \subseteq \alpha$, $x : r \in \Delta$, and (1) the subtrees T_1 and T_2 do not contain any application of the mingle rule, (2) there is no application of restart using a query which is an ancestor of

$$\Gamma \vdash^{?} \alpha : q, H * (\Gamma, \alpha, q).$$

Let $\alpha = \beta' \cup \gamma$. Then, we obtain a transformed derivation $\mathcal{D}'$ from $\mathcal{D}$ as shown in Fig.5.5. In the shown derivation, T_1' is obtained from T_1 by deleting β' on any node on the path in T_1 leading from the root (the query $\Gamma \vdash^{?} \gamma : q, H * (\Gamma, \gamma, q)$) to

$$\Delta \vdash^{?} \beta : r, H * (\Gamma, \alpha, q) * H_1 * (\Delta, \beta, r)$$

which now becomes

$$N = \Delta \vdash^{?} \phi : r, H * (\Gamma, \gamma, q) * H_1' * (\Delta, \phi, r),$$

where $\phi = x$ or $\phi = \emptyset$. (H_1' is obtained from H_1, by deleting β'). Notice that, by the splitting condition in the reduction rule, β can only be on that path. We know that $x : r \in \Delta$, and hence N succeeds. The

$$\begin{tikzcd}
 & K \arrow[dl] \arrow[d] \arrow[dr] & \\
\dots & \vdots & \dots \\
 & \Gamma \vdash^? \alpha : q, H * (\Gamma, \alpha, q) & \\
 & T_1 & \\
 & \Delta \vdash^? \beta : r, H * (\Gamma, \alpha, q) * H_1 * (\Delta, \beta, r) & \\
 & \Gamma \vdash^? \beta' : q, H * (\Gamma, \beta', q) & \\
 & T_2 &
\end{tikzcd}$$

Figure 5.4:

$$\begin{tikzcd}
 & K \arrow[dl] \arrow[d] \arrow[dr] & \\
\dots & \vdots & \dots \\
 & \Gamma \vdash^? \alpha : q, H * (\Gamma, \alpha, q) \arrow[dl] \arrow[dr] & \\
\Gamma \vdash^? \beta' : q, H * (\Gamma, \beta', q) & & \Gamma \vdash^? \gamma : q, H * (\Gamma, \gamma, q) \\
T_2 & & T_1'
\end{tikzcd}$$

Figure 5.5:

subderivation T_1' differs from T_1 only for the path from the root that is now

$$\Gamma \vdash^? \gamma : q, H * (\Gamma, \gamma, q)$$

to N it is easy to see (by proposition 5.8.3) that if T_1 is successful, then so is T_1'. We can therefore conclude that if $\mathcal{D}$ is successful, so is $\mathcal{D}'$. But $\mathcal{D}'$ contains a minimal application of restart less than $\mathcal{D}$.
□

By the previous results, restart and mingle are equivalent. We now show the soundness and completeness of the procedure P1 which makes use of mingle rule, because is technically simpler. In P1, we no longer need to record the history of the computation, thus we will drop the parameter H from queries.

We adopt a semantics for **RM0**similar Fine's semantics of section 5.5.1. In addition to the condition of definition 5.5.3 (1) - (6) satisfied by R, we postulate, that

(*) $x \leq x \circ x$

and that the evaluation fuction satisfies

$V(x) \cap V(y) \subseteq V(x \circ y)$.

Since $\leq$ is a paratial ordering, the condition (5) $x \circ x \leq x$ and (*) make $\circ$ an idempotent operation which can be thought as a semilattice operation $\cup$ with identity 0. However neither $x \leq x \cup y$, nor $x \cup y \leq x$ is assumed.

Definition 5.8.7 A model structure M for $\mathcal{L}(\rightarrow)$ is a tuple

$$M = (S, \cup, 0, V),$$

where

- $(S, \cup, 0)$ is a semi-lattice with zero (the element 0), and
- V is a function of type $S \rightarrow Pow(Var)$, satisfying the following condition (M): for all $p \in Var$, $x, y \in S$
$$p \in V(x) \wedge p \in V(y) \rightarrow p \in V(x \cup y).$$

Truth and validity are defined as in the case of **R**, we refer to section **??**. ■

We notice that condition (M) implies that for any formula A,

$M, x \models A$ and $M, y \models A$ implies $M, x \cup y \models A$.

We we only sketch the proofs of the and completeness. With respect to soundness, we observe that we have to state it in a more general form, since the label x of some formula $x : A \in \Delta$ actually used in a derivation of $\Delta \vdash^? \alpha : G$ need not be in α.

Theorem 5.8.8 *Let Γ be a database, for any α and A, if $\Gamma \vdash^? \alpha : A$ succeeds in P1, then there is a database $\Delta \subseteq \Gamma$ such that*

(a) $\Delta \vdash^? \lambda_\Delta : A$ succeeds;

(b) $\alpha \subseteq \lambda_\Delta \subseteq \lambda_\Gamma$;

(c) letting $\Delta = \{y_1 : B_1, \ldots, y_k : B_k\}$, we have that

$$B_1 \rightarrow B_2 \rightarrow \ldots \rightarrow B_k \rightarrow A \textit{ is valid in RM0.}$$

Proof. By induction on the heigth h of a successful derivation of $\Gamma \vdash^? \alpha : A$. We sketch the case of reduction and Mingle.

(Reduction) Let $h > 0$ and $A = q$ is an atom. Suppose that the reduction rule is applied, then for some $y : C \in \Delta$ with $C : A_1 \rightarrow \ldots \rightarrow A_k \rightarrow q$, from $\Delta \vdash^? \alpha : q$, we step to

$$\Delta \vdash^? \alpha_i : A_i,$$

which succeed with height $h_i < h$, for $i = 1, \ldots k$, and it holds that:

1. $\alpha_i \cap \alpha_j = \emptyset$, for $i \neq j$;
2. $\bigcup_{i=1}^k \alpha_i = \alpha - \{y\}$.

By induction hypothesis, there are $\Delta_i \subseteq \Gamma$, for $i = 1, \ldots, k$, such that letting $\lambda_{\Delta_i} = \gamma_i$, we have:

$\Delta_i \vdash^? \gamma_i : A_i$ succeeds with height $h'_i \leq h_i$, and
$\alpha_i \subseteq \gamma_i \subseteq \lambda_\Gamma$,

and assuming $\Delta_i = \{x_{i,1} : C_{i,1}, \ldots, x_{i,r_i} : C_{i,r_i}\}$, we have:

$\models C_{i,1} \to \ldots \to C_{i,r_i} \to A_i$.

We can define

- $\Delta = \bigcup_i \Delta_i \cup \{y : C\}$, and,
- $\delta_i = (\gamma_i - \{y\}) - \cup_{j=i+1}^k \gamma_j$, for $i = 1, \ldots, k$
- $\gamma = \bigcup_i \delta_i \cup \{y\}$.

It is immediate to check that (1) $y \notin \delta_i$, (2) $\gamma = \cup_i \gamma_i \cup \{y\}$, (3) $\delta_i \subseteq \gamma_i$, (4) $\delta_i \cap \delta_j = \emptyset$, (5) $\bigcup_i \delta_i = \gamma - \{y\}$. Since $\Delta_i \subseteq \Delta$, and (3), we have $\Delta \vdash^? \delta_i : A_i$ succeeds with height $h''_i \leq h'_i$, so that by (4) and (5), by reduction rule we can conclude that $\Delta \vdash^? \gamma : q$, succeeds. Since $\Delta_i \subseteq \Gamma$, and $y : C \in \Gamma$, we have $\Delta \subseteq \Gamma$. It is also clear that $\gamma = \lambda_\Delta \subseteq \lambda_\Gamma$. We have still to check that $\alpha \subseteq \gamma$. It holds $\cup_i \alpha_i = \alpha - \{y\}$, so that we have:

$$\alpha \subseteq \bigcup_i \alpha_i \cup \{y\} \subseteq \bigcup_i \gamma_i \cup \{y\} = \gamma.$$

This concludes (a) and (b).

(Part (c)). We know that for $i = 1, \ldots, k$

$$(i) \models C_{i,1} \to \ldots \to C_{i,r_i} \to A_i.$$

If $y : C \notin \bigcup \Delta_i$, we show that

$E = C_{1,1} \to \ldots \to C_{1,r_1} \to \ldots \to C_{k,1} \to \ldots \to C_{k,r_k} \to C \to q$ is valid.

Suppose it is not. Then there is a model $M = (S, \cup, 0, V)$ such that $M, 0 \not\models E$ then, there are $x_{1,1}, \ldots, x_{1,r1}, \ldots, x_{k,r_k}, y \in S$, such that

$$(ii)\ M, x_{i,j} \models C_{i,j},$$

for $j = 1, \ldots, r_i$, $i = 1, \ldots, k$, and

$$(iii)\ M, y \models A_1 \to \ldots \to A_k \to q,$$

and, letting $s_i = x_{i,1} \cup \ldots \cup x_{i,r_i}$ it holds that: $M, s_1 \cup \ldots \cup s_k \cup y \not\models q$. By (i) and (ii) we can conclude that

$$(iv)\ M, s_i \models A_i.$$

By a simple inductive argument it is easy to see, using (iii) and (iv) that for $i = 1, \ldots, k-1$ we get:

$M, s_1 \cup \ldots \cup s_i \cup y \models A_{i+1} \to \ldots \to A_k \to q$,

so that we finally get $M, s_1 \cup \ldots \cup s_k \cup y \models q$. against the hypothesis.

If $y : C \in \bigcup \Delta_i$, we show in a similar way that

$$C_{1,1} \to \ldots \to C_{1,r_1} \to \ldots \to C_{k,1} \to \ldots \to C_{k,r_k} \to q,$$

is valid.

(Mingle) Suppose that the mingle rule is applied to $\Gamma \vdash^? \alpha : q$ so that we step to

$$(*)\Gamma \vdash^? \alpha_1 : q \quad \text{and} \quad \Gamma \vdash^? \alpha_2 : q,$$

with $\alpha_1 \cup \alpha_2 = \alpha$. Since (*) succeeds with height $h_i < h$, we can apply the inductive hypothesis and obtain that there are Δ_i (for $i = 1, 2$) such that $\Delta_i \subseteq \Gamma$, $\alpha_i \subseteq \lambda_{\Delta_i} \subseteq \lambda_\Gamma$, $\Delta_i \vdash^? \lambda_{\Delta_i} : q$ succeeds with $h'_i \leq h_i$, and letting $\Delta_i = \{A_{i,1}, \ldots, A_{i,k_i}\}$, we have that in **RM0**is valid

$$A_{i,1} \to \ldots \to A_{i,k_i} \to q.$$

We let $\Delta = \Delta_1 \cup \Delta_2 = \{A_{1,1}, \ldots, A_{1,k_1}, A_{2,1}, \ldots, A_{2,k_2}\}$, so that we easily have that $\alpha \subseteq \lambda_\Delta \subseteq \lambda_\Gamma$, and

$A_{1,1} \to \ldots \to A_{2,k_1} \to A_{2,1} \to \ldots \to A_{2,k_2} \to q$ is valid in **RM0**,

and by Propositions 5.8.2, 5.8.3 the two queries

$\Delta \vdash^? \lambda_{\Delta_1} : q$ and $\Delta \vdash^? (\lambda_{\Delta_2} - \lambda_{\Delta_1}) : q$ succeed.

Thus, from $\Delta \vdash^? \lambda_\Delta : q$, we can step to the above two queries and succeed. $\square$

Completeness can proved by showing first the admissibility of the cut-rule and then by providing a canonical model construction. The proof of this theorem is similar to the one of theorem **??**, but acutally simpler.

Theorem 5.8.9 *Let* $\Gamma = \{x_1 : A_1, \ldots, x_n : A_n\}$*, and* $\Delta = \{y_1 : B_1, \ldots, y_k : B_n\}$*, and let* $\alpha = \lambda_\Gamma$*,* $\beta = \lambda_\Delta$*,* $u \notin \alpha \cup \beta$*, if*

$\Gamma[u : C] \vdash^? \alpha : D$ *and* $\Sigma \vdash^? \beta : C$ *succeed in P1*

then also

$\Gamma[u/\Sigma] \vdash^? \alpha[u/\beta] : D$ *succeeds in P1.*

The canonical model $M = (W, \cup, \emptyset, V)$ is defined as follows: $\cup$ is set-union, $\emptyset$ is the emptyset, and V is the evaluation function of type $W \to Pow(Var)$, defined by the condition below: for $\Delta \in W$,

$p \in I(\Delta) \quad \Leftrightarrow \quad \Delta \vdash^? \lambda_\Delta : p.$

It is easy to see that M satisfies all the conditions of definition **??** and that for every formula C:

$$M, \Delta \models C \quad \text{iff} \quad \Delta \vdash^? \lambda_\Delta : C \text{ succeeds},$$

from which the completeness easily follows.

Theorem 5.8.10 *If A is valid in RM0, then* $\emptyset \vdash^? \emptyset : A$ *succeeds in P1.*

At the beginning of the section, we have seen that *self restart* is needed to consume several copies of an atomic formula which immediately succeeds. Self-restart can be avoided if we change the label discipline of formulas. It is easy to see that if a formula occurs in a database, its number of copies is immaterial, in the sense that:

$$(1)\ \Delta, x : A \vdash^? \alpha \cup \{x\} : B, H \text{ succeeds iff}$$

$$(2)\ \Delta, x_1 : A, \ldots, x_k : A \vdash^? \alpha \cup \{x_1, \ldots x_k\} : B, H \text{ succeeds,}$$

(it is $k > 0$). To see this, take a successful derivation of (1), this could be also a derivation of (2), apart from some copies of A (witnessed by their labels) which remain unused in some node; in order to consume them we restart and attach below any such node another copy of the derivation of (1). Self restart is needed to deal with the above situation when $B = A$ is atomic, whence the query immediately succeeds. The above fact means that we can economize labels and eliminate self restart by labeling with the same label all copies of A in the database. That is, a database contains *at most one* occurrence of any formula. To this purpose we first modify the implication rule, so that we no longer introduce a formula in the database if it already occurs in it. We can also add a loop-checking mechanism, as defined in section 5.7 and make it terminating. We leave to the reader to work out the details of such a procedure.

5.9 Relation with other approaches

WHAT SHALL WE PUT IN THIS SECTION???

Relevance logics In principle a proof procedure for relevance logic (at least for R witout distribution and with an intuitionistic negation [**?**]) can be easily obtained: adopt any proof- method for intuitionistic logic adding a mechanism to check the usage of formulas. This checking can be done by marking formulas as long as they are used. Every time we use a formula we mark it; if the proof succeeds according to the method for intuitionistic logic, we check that every formula in the database has been marked. If it is so, we succeed, otherwise we fail. This idea has been employed in [McRobbie and Belnap 79] to define tableau procedures for many relevance logics. If we adopt this strategy in a goal-directed procedure, we encounter a problem. A derivation may split in branches or subderivations. A formula might be used (and marked) in one branch but not in another. Therefore, we cannot say whether a query in a derivation tree is successful or not, unless we inspect the whole derivation tree. This situation is not very very satisfactory: whether a query in a leaf of a derivatoin tree is to be regarded as successful or not, will come to depend on what there is on other branches of the derivation. We would like instead that $\Gamma \vdash^? A$ being provable or not only depends on the subtree whose root is $\Gamma \vdash^? A$. Bollen has developed a goal-directed procedure for a fragment of **R** [Bollen 91]), which does not have this problem. His idea is to avoid splitting derivations in branches. We can formulate a proof procedure, in which we maintain a global proof-state ($\Delta \vdash^? [A_1, \ldots, A_n]$), where all $A_1, \ldots A_n$ have to be proved (they can be thought as linked by a *relevant* conjunction). For example, from

$$a \to b \to c, a, b \vdash^? c,$$

we step to

$$(a \to b \to c)^*, a, b \vdash^? a, b$$

and we succeed by marking both a and b

$$(a \to b \to c)^*, a^*, b^* \vdash^? a, b.$$

However, if we want to keep all subgoals together, we must take care that different subgoals A_i may happen to be evaluated in different contexts. For instance in

$$C \vdash^? D \to E, F \to G,$$

according to the implication rule, we must evaluate E from $\{C, D\}$, and G from $\{C, F\}$. Bollen accommodates this kind of context-dependency by indexing subgoals with a number which refers to the part of the database that can be used to prove it. More precisely, in the implicational fragment, the database is regarded as a tree of formulas (corresponding to the nesting of subderivations), and the index of each goal denotes the path of formulas in the database tree which can be used to prove the goal. Furthermore, a list of numbers is maintained to remember the usage; in a successful derivation the usage list must contain the numbers of all formulas of the database.The whole mechanism, though efficient, is not very terse from the proof-theoretical perspective. However, Bollen's proof system is more extended than ours, it is a logic programming language and it is defined for a fragment of first-order **R**.

Linear logic

Many people have developed goal-directed procedures for fragments of linear logic, leading to the definition of logic programming languages based on linear logic. We notice that relevant logic can be encoded in linear logic by replacing $A \rightarrow B$ by $!(A - \circ B)$. The several proposals differ in the fragment of the language they chose as primitive. Much emphasis is given to the treatment of the operator !, the exponential operator, which is needed for having a meaniningful language (we want some resources to be permanent). Some proposals as [**?**] and [**?**] take as basis the multiple sequent version of linear logic. We notice, however, that the exponential-free (propositional) fragment of linear logic is perhaps the simplest of the substructural logics. A detailed comparison is out of the scope of the present chapter.

Other Approaches - APPROACHES BASED ON DISPLAY LOGIC
- LABELLED CALCULI (Gabbay, D'Agostino, Russo and Broda, Vigano et al.)

Chapter 6

Conclusions an Further work

6.1 A procedural interpreation of logics

THIS SECTION SHOULD BE REVISED I AM NO LONGER CONVINCED ABOUT THE THREE STAGE PRESENTATION OF A LOGIC. IN PARTICULAR, THE CONCEPT OF ALGORITHMIC PROOF SYSTEM IS TOO GENERAL, ANY PRESENTATION OF A LOGIC IS ALGORITHMIC (IN THE SENSE OF BEING R.E.) IF THE LOGIC IS R.E. ITSELF!

This book underlies a procedural interpretation of logical systems. In the traditional view, a logical system can be mathematically presented either as a consequence relation satisfying certain properties, or as a set of axioms in some language, or, finally, through its semantics, as a class of mathematical structures. Usually one starts with such a mathematical presentation of a system and later on studies the problem of deduction in that logical system. Suppose we have fixed a logical system S, with its mathematical presentation and we have defined some notion of consequence relation relative to S. Here by consequence relation we simply mean a notion of "following according to S", so that we can make statements of the form "information (formula) A follows from information (data) Δ according to S", denoted by

$$\Delta \vdash_S A.$$

The consequence relation $\vdash_S$ is expected to satisfy certain properties to be considered a logic, namely, reflexivity, transitivity and cut.

It is convenient to refer to the presentation of a system as described above as the *mathematical stage* in defining a logic. We mean that $\vdash_S$ is defined *mathematically*, but not necessarily *algorithmically*. We do not necessarily provide an algorithm to check given Δ and A whether $\Delta \vdash A$ holds or not, nor can we necessarily recursively generate all pairs (Δ, A) for which $\Delta \vdash A$ holds. For example, how do we check, in the case of intuitionistic logic, presented through its Kripke semantics, whether:

$$\{A, ((B \to A) \to B) \to B\} \vdash^? B$$

We have no algorithm to use.

We regard the second stage in the presentation of a logic as the *algorithmic proof* stage. This means that we actually have an algorithm for generating pairs (Δ, A) such that $\Delta \vdash A$ holds. We mean here an algorithm in the mathematical sense, i.e. some given procedure for checking whether $\Delta \vdash A$ holds. In logical terms this means that the set of pairs (Δ, Q) such that $\Delta \vdash Q$, is recursively enumerable. It need not be a practical automated algorithm which is actually implemented. So, for example, a recursive function generating the pairs (Δ, Q) such that $\Delta \vdash Q$, is a mathematical algorithm in our sense. On the other hand, a program which implements a decision procedure for checking whether $\Delta \vdash^? Q$ holds is the automated practical algorithm. It is quite possible that the set of pairs (Δ, Q) such that $\Delta \vdash Q$, is actually recursive. For reasons of convenience the system may initially be presented as an algorithmic proof system where it is not immediately clear that it is indeed recursive. In such a case we may obtain an effective decision procedure from the algorithmic proof system via optimization. Of course some logics can be presented directly in their algorithmic stage. We can give a recipe for checking whether $\Delta \vdash^? A$ holds or not, and provided we show that the relation: 'The procedure for checking $\Delta \vdash^? A$ terminates with answer yes' is a consequence relation, we have properly defined a logic.

For some logics $\vdash$ (i.e. consequence relations) which can be defined mathematically, there are no algorithms for enumerating the cases when $\Delta \vdash A$ holds. So, for these non-recursively enumerable (some are even not arithmetical) logics there is no algorithmic stage (there may be though a *transfinite* proof procedure).

What about the automated stage? This stage is a practical implementation of an algorithmic stage. It should be sound (i.e. if the automated stage says that $\Delta \vdash A$ should hold then it does indeed hold) and possibly complete (i.e. if $\Delta \vdash A$ does hold then the automated stage can confirm that). We say 'possibly' complete because we do not necessarily require completeness. We can regard the algorithmic stage as a *proof checker* stage and the automated stage as a *proof finder* stage. The reason for this terminology is that an algorithmic proof system is generally recursive enumerable and can effectively be used in general mainly to verify, for a given proof, if it is correct in the system. On the other hand, an effective (recursive in polynomial time on average) proof system can be used for finding proofs.

Here is how these stages might look for the case of classical logic.

Mathematical stage: $\Delta \vdash A$ is defined according to the classical truth tables, in the usual traditional manner.

Algorithmic stage: A Gentzen system for classical implication.

Automated stage: A machine implementation of the Gentzen system.

Note that in this example the truth table method can serve as the algorithmic stage as well.

For any one consequence relation there can be more than one mathematical presentation and many algorithmic presentations. Each algorithmic presentation may have many machine implementations.

The theme of the three-stage presentation of logics is important not only from the classification point of view but also from the theoretical point of view.

Consider an algorithmic proof system for a logic **L1**. Let us call it **S1**. Thus whenever $\Delta \vdash_{\mathbf{L1}} A$ holds, the algorithmic procedure **S1** would succeed when applied to $\Delta \vdash^? A$. **S1** contains manipulative rules. These rules can be tinkered with, changed, modified and made more efficient. It happens in many practical cases that by making *natural* changes in **S1** we get a new algorithmic system **S2** which defines a new logic **L2**. **L2** may be a well- known logic already mathematically defined, with completely different motivation. The insight that **S2**, the result of tinkering with **S1**, is an algorithmic system for **L2** can deepen our understanding of **L2**.

We thus can obtain a network of logics interconnected on many levels, mathematical and algorithmic, where different logics can be obtained in different ways from other logics in the network by making some *natural* changes.

Thus our view of logic is procedural. The declarative nature is only a component in the formulation of the logic. This is a somewhat departure from the traditional point of view. We intuitively define the notion of a recursively enumerable logical system $\mathbf{L}$ as a pair $\mathbf{L} = (\vdash, \mathbf{S})$, where $\vdash$ is a mathematically defined consequence relation and $\mathbf{S}$ is an algorithmic proof system for $\vdash$. The algorithmic system is sound and complete for $\vdash$. Thus different algorithmic systems for the same $\vdash$ give rise to different logics, according to our definition.

To make this new notion more intuitively acceptable, consider the following example. Take a Gentzen style formulation for intuitionistic logic. A minor change in the rules will yield classical logic. Another minor change will yield linear logic. Thus from the point of view of the algorithmic proof system (i.e. Gentzen proof theory) linear logic, intuitionistic logic and classical logic are neighbours or brother and sister.

Now consider classical logic from the point of view of the two-valued truth table. It is easy to generalize from two-values to Łukasiewicz n-valued logic L_n. From the truth table point of view, classical logic and L_n are neighbours. However, there is no natural Gentzen formulation for L_n and so it cannot be directly related to intuitionistic logic.

Now consider a Hilbert style presentation of classical logic, intuitionistic logic and Łukasiewicz n-valued logics. Such axiomatizations exist. Through the Hilbert presentation, the relationship between the three systems is very clear. Some axioms are dropped and/or added from one to obtain the other.

Our view is that different algorithmic proof presentations of a logic (in the old sense, i.e. the set of theorems) gives us different logics (in the new sense). Thus we have three distinct logics (all versions of classical logic) namely:

- Classical logic truth table formulation;
- Classical logic Hilbert system formulation;
- Classical logic Gentzen formulation.

These are different also from the point of view of the kind of information on the logic they highlight. A Gentzen system and a Hilbert system can give us a rather different type and style of information about classical logic (old sense). The same can be said for other logics. We can look at different styles of algorithmic systems as (algorithmic) proof methodologies. This book introduce another style of presentation of logics, the goal-directed one. We can re-interpret old concepts and distinctions in this new perspective. To make the argument more concrete let us discuss an example at from chapter 2.

The difference between classical implicational logic and intutionistic implicational logic boils down to the fact that the first allows unbounded restart, whereas the latter only bounded restart. As an immediate consequence backtracking is necessary for intutionistic logic, whereas it is not needed for classical logic (see chapter 2). In our opinion, this gives a very intuitive and "computational" picture of their difference even from the point of view of complexity.

On the other hand, their different behaviour with respect to the goal-directed computation may also be "explained" by mapping goal-directed derivations into derivations in a multiple-conclusion sequent calculus. We can notice that unbounded restart corresponds to keep side formulas in the conclusions even when we change context, and we are not allowed to do so in case of intutiotionistic logic[1].

[1]Moreover the restrictions we must put on the rules in the case of intuitionistic logic causes the loss of permutability of derivation steps. This latter observation "explain" the need of backtracking in terms of sequent calculi. Of course a similar explanation works directly in the goal-directed case and it is made precise by proposition ???.

Although this type of analysis is significant and we intend to carry it on, as we explain the next section, we do not think that it conveys a deeper "understanding" of a logical system than the goal-directed presentation itself. It just another presentation and it is useful to observe the same object from different perspectives.

6.2 Future Work

There are a number of issues which deserve to be studied further. We list a number of open problems which will be addressed in future work.

Relation with other proof systems
It is important to compare the goal-directed methodology with other methodologies to the purpose of understanding the relative merits and de-merits of each one. For calculi for which there are well-known proof-systems (like sequent of tableaux) the purpose is to see how goal-directed analysis helps in proof-search for these systems. To this aim we should be able to define a formal mapping between goal-directed proofs and and proofs with the other systems. For other systems for which the proof-theory is not well-understood yet, the goal directed approach can suggest how to develop traditional proof methods such as tableaux or sequent calculi (it is the case of strict implication modal logics, their intuitionistic variants and of intermediate logics BH_n). Some natural concepts in the context goal-directed computation seems to have a relation, but they do not have an exact counterpart in other systems. This is the case of the restart rule. Another example is the notion of "diminishing resource" computation. The latter corresponds to some limitation of duplication or contraction in sequent systems, but the exact corresponding notion in other proof systems has still to be found.

Interpolation and other properties
We can use goal-directed methods to prove and define properties of logics; To this regard, we have seen in Chapter 2 that we can constructively prove an interpolation property for intitionitic implicational logic. Can a corresponding interpolation property be obtained in a similar way for other logical systems? An important property of the goal-directed proof procedures is that the cut-rule, suitably formulated, is admissible. This property has the same import as cut-elimination property in Gentzen System. It would be interesting to calculate an upper bound on the cut-elimination process and compare it with what is known in case of sequent calculi.

Decidability and Complexity bounds
Goal-directed proof procedures help in proof-search. The search of a proof is driven by the goal and only when we reach atomic constistuents we really have a non-deterministic step (reduction or restart). However, the simple goal-directed proof-procedures are subject to two problems: the first one is the possible non-termination and the other is that the computation may contain repeated subproofs. We have already remarked upon the first problem. In this book we have exemplified two methodologies to deal with non termination: the first one is to adopt "diminishing resource policy", the other is to augment the system by a loop-checking mechanism. We experimented both strategies for intuitionistic logic. There are logics which are known to be decidable for which we have not devised yet any mechanism of the either two kinds. This is the important case, for instance, of the modal strict implication logics. Once that we have a terminating procedure we can ask what optimizations we can make. This can lead to improvements in both space and time complexity. In [?] it is shown that a simple refinment of the goal-directed procedure produce an $n\ logn$-space procedure for intuitionistic logic. We hope to develop

a similar analysis further.

One way of handling the problem of repeated subproofs is to store every partial results, this correspond to keep a sort of a table of intermediate results and correpond to apply a form of analytic cut:

from $\Gamma, A \to B \to q \vdash q$, we step to $\Gamma \vdash A$ and $\Gamma, A \vdash B$.

That is to say, we remember when we try B (the second subgoal) that A has already succeeded.

Proof-extraction
The goal-directed procedures can be used to show type-inhabitation, see [**?**] for the case of intutionistic logic. We wonder whether one can use similar algoritm for other logics. It might be also worth investigating if, by means of Curry-Howard isomorphism, there class of λ-terms which represent goal-directed proofs can be characterized in any way.

Extensions of the language
We have concentrated on the implicational fragment. For the case of intutionistic logic, we have extended our procuderes to the full propositional calculus and to the $\forall, \to$ fragment, can we make the same extensions for other systems? Here there are a number of problems. It must be noticed that we could obtain an extension with disjunction for intutionistic logic only by making use of labels. Thus, we are expected to need such machinery for other logics. The other problem is that in some cases (such as the one of relevance logics) the mere addition of the usual *distributive* lattice connectives makes the respective systems undecidable (actuallt **T**is not known to be decidable even in its pure implicational fragment). Thus, we do not expect the extension be easy and computationally favourable.

Starting with the implicational fragment, we have seen that one can easily extend the methods to a broader class of formulas, for instance Hereditary-Harrop formulas. On the other hand, it might be interesting to study subclasses of the implicational fragment, for instance to isolate a notion of Horn database and optimize our procedures for this case.

In case of modal logic, the most prominent extension is to allow in the database formulas with $\Diamond$-operator in their head. This in not expected to be easy, one has similar problem as the one of allowing existentially quantified data in the database.

Another prominent extension is the one to first-order languages. Following the line of logic programming (as we did in chapter 2, for intutionistic logic), we want to get mechanisms which actually compute answer-substituions. In general in most non-classical logics there are several options in the interpretation of quantifiers and terms according to the intendended semantics (tipically one several options: constant-domain, increasing domains etc.); moroever one may adopt either rigid, or non-rigid interpretation of terms. It is likely that in order to develop the first-order extensions we will need to label the data as we did in several cases. We would like to represent the semantic options we have mentioned by tinkering the unification and the skolemization mechanism. The restrictions on unification and skolemization might depend on the labels associated with the data. The treatment of $\forall - \to$-fragment will be the initial objective.

Deductive databases in NonClassical Logics
The idea is to use non-classical logics as a representation language and implement a goal-directed procedure as a conventional query-answering mechanism. Of course this enterprise presupposes we have in mind some areas in which the non-classical logics we have studied in this book may find an application. However, the entire battery of logic programming concepts and methods may be imported

in our goal-directed procedures. For instance one can define non-monotonic extensions of the basic procedures, such as the concept of negation by failure. One can also define abductive extensions of the basic proof-mechanism to compute abductive explanations of given data. Examples of developed systems so far includes the use of intuitionistic logic, classical, linear, relevant, modal and multimodal S4. In this respect our proof procedures might be considered as "abstract inference engines" for implementing some non-classical logic programing languages. The usefulness of such non-classical logic programming languages (either extensions or alternatives to classical Horn one) as specification and representation languages has not fully been investigated yet, with the exception of intutionistic logic, linear logic and temporal logic. We believe that these examples do not exhaust the interest in developing non-classical logic programming languages.

Extension to other Logics
Although our methods seem to cover a broad family of logics there are some important holes in the landscape of logics which we have been able to treat. More than capturing specific (and exotic) systems, *families* of logics are important here. Perhaps the most important class of logic which we have been unable to capture so far is the class of many-valued logics. The only exception has been Dummett's logic LC, that has we have seen in Chapter 3, can be interpreted as many-valued logic. All the other systems, notably the infinite-vaued ones such as Łukasiewicz and product logic (which are consider together with LC the main fuzzy logics, see[Hajek 98]) are out of our reach for the moment.

There is a much work to be done in this respct also on extensions of classical logic. For instance multi-modal logic and contitional logics which have recently received a strong interest for their application in a numebr of areas: reasoning about actions, distributed knowledge, and agent communication (multi-modal logics), hypothetical, couterfactual reasoning, reasoning about belief change and theory update (conditional logics). All these systems are worth studying for their goal-directed formulation.

The above list of topics is by no means concluded. We plan to carry on this research in a latter volume.

Bibliography

[Abadi and Manna 86] M. Abadi, Z. Manna, Modal Theorem Proving *in* Proc. 8th Int Conf. on Automated Deduction, LNCS 230, Springer Verlag, pp. 172 - 189, 1986.

[Abadi and Manna 89] M. Abadi, Z. Manna, Temporal Logic Programming, *J. Symbolic Computation*, 8: 277-295 (1989).

[Avellone et al.98] A. Avellone, M. Ferrari, P. Miglioli, Duplication-free tableau calculi together with cut-free and contraction-free sequent calculi for the interpolable propositional intermediate logics, *J. of Logic and Computation*, to appear, 1998.

[Anderson and Belnap 75] A.R. Anderson, N.D. Belnap, Entailment, The logic of Relevance and Necessity, Vol.1, Princeton University Press, New Jersey, 1975.

[Anderson et al. 92] A.R. Anderson, N.D. Belnap, J.M. Dunn, Entailment, The logic of Relevance and Necessity, Vol.2, Princeton University Press, New Jersey, 1992.

[Avron 87] A. Avron, A constructive analysis of **RM**, *J. Symbolic Logic*, 52: 277-295 (1987).

[Avron 91a] A. Avron, Hypersequents, logical consequence and intermediate logics for concurrency, *Annals of Mathematics and Artificial Intelligence*, 4: 225-248 (1991).

[Avron 91b] A. Avron, Simple consequence relations, *J. Logic and Computation*, 2: 105-139 (1991).

[Baaz and Fermüller 99] M. Baaz, C. Fermüller Analytic Calculi for Projective Logics, *in* Proc. of TABLEAUX99, LNCS, to appear, 1999.

[Balbiani and Herzig 94] P. Balbiani, A. Herzig A translation from modal logic G into K4, *J. Applied Non-Classical Logics*, 4(1): 73-77 (1994).

[Basin et al. 97a] D. Basin, S. Matthews, L. Viganò. Labelled propositional modal logics: Theory and Practice. *J. of Logic and Computation*, 7(6): 685-717, 1997.

[Basin et al. 97b] D. Basin, S. Matthews, L. Viganò. A new method for bounding the complexity of modal logics. *In* Proc. of Fifth Gödel Colloquim, LNCS 1289, pp 89-102, 1997.

[Basin et al. 99] D. Basin, S. Matthews, L. Viganò. Natural deduction for non-classical logics. *Studia Logica*, 60(1): 119-160, 1998.

[Bollen 91] A. W. Bollen, Relevant Logic Programming, *J. Automated Reasoning*, 7: 563-585 (1991).

[Boolos 79] G. Boolos, The Unprovability of Consistency - an essay in modal logic, Cambridge University Press, 1979.

[Cerrato 93] C. Cerrato, Modal sequents for normal modal logics, *Mathematical Logic Quarterly*, 39: 231-240 (1993).

[Cerrato 94] C. Cerrato, Natural Deduction based upon strict implication for normal modal logics, *Notre Dame J. of Formal logic*, 35(4): 471-495 (1994).

[Ciabattoni et al.] A. Ciabattoni, D. Gabbay, and N. Olivetti. Cut-free proof systems for logics of weak excluded middle. appariråsu *Soft Computing*,

[Corsi 87] G. Corsi, Weak logics with strict implication, *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 33: 389-406 (1987).

[Dyckhoff 92] R. Dyckhoff, Contraction-free sequent calculi for intuitionistic logic, *J. Symbolic Logic*, 57: 795-807 (1992).

[Dickhoff 98] R. Dyckhoff, A deterministic terminating sequent calculus for propositional Dummett logic, draft, abstract for Logic Colloquium 98, 15 April, 1998.

[D'Agostino and Gabbay 94] M. D'Agostino, D. Gabbay, A generalization of analytic deduction via Labelled Deductive Systems I: Basic Substructural Logics, *J. Automated Reasoning*, to appear.

[Dosen 85] K. Dosen Sequent systems for modal logic, *Journal of Symbolic Logic*, 50:149-168 (1985).

[Dummett 59] M. Dummet A propositional calculus with denumerable matrix, *Journal of Symbolic Logic*, 24:96-107 (1959).

[Dung 91] P. M. Dung. Negations as hypotheses: an abductive foundation for logic programming. In *Proc. ICLP-91 Conference*, pages 3–17, 1991.

[Dung 93] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming. In *Proc. IJCAI93*, pages 852–857, 1993.

[Dunn 86] J. M. Dunn Relevance logic and entailment, *in* (D. Gabbay and F. Guenthner eds.) Handbook of Philosophical logic, vol III, pp. 117-224, D. Reidel Publishing, Dordercht,1986.

[Eshghi89] K. Eshghi and R. Kowalski. Abduction compared with negation by failure. In *Proc. 6th ICLP*, pages 234–254, Lisbon, 1989.

[Ewald 86] W. B. Ewald Intuitionistic tense and modal logic, *J. Symbolic Logic*, 51(1): 166-179, 1986.

[Enjalbert and Farinãs 89] P. Enjalbert and L. Farinãs del Cerro, Modal resolution in clausal form, *Theoretical Computer Science*, 65: 1-33, 1989.

[Farinãs 86] L. Farinãs del Cerro MOLOG: a system that extends Prolog with modal logic, *New Generation Computing*, 4:35-50 (1986).

[Fitting 83] M. Fitting Proof methods for Modal and Intuitionistic Logic, vol 169 of Synthese library, D. Reidel, Dorderecht, 1983.

[Fitting 90] M. Fitting Destructive modal resolution, *J. Logic and Computation*, 1(1): 83-98 (1990).

[Gabbay 84] D. M. Gabbay, Lecture notes on *Elementary Logic - A Proc edural Perspective*, Imperial College, 1984.

[Gabbay 85] D. M. Gabbay, N-Prolog Part 2, *Journal of Logic Programming*, 251–283, 1985.

[Gabbay 87] D. M. Gabbay, Modal and Temporal Logic Programming, *in* (A. Galton ed.) Temporal logics and their applications, Academic Press, 1987.

[Gabbay 91] D. M. Gabbay, Algorithmic proof with diminishing resources. *in* Proc. of CSL'90, LNCS vol 533, , Springer-Verlag, pp.156-173, 1991.

[Gabbay 92] D. M. Gabbay, Elements of Algorithmic Proof, *in* (S. Abramsky, D. M. Gabbay, T. S. E. Maibaum eds.), Handbook of Logic in Theoretical Computer Science, Oxford University Press, 1992.

[Gabbay 93] D. M. Gabbay, General theory of structured consequence relations, *in* (P. Schroeder-Heister, K. Dosen eds.) Substructural logics, Studies in Logic and Computation, Oxford University Press, pp. 109-151, 1993.

[Gabbay 96] D. M. Gabbay, Fibred Semantics and the Weaving of Logics. Part I: modal and intuitioistic logic, *J. Symbolic Logic*, 61:1057–1120, 1996.

[Gabbay 98] D. M. Gabbay, *Elementary Logic*, Prentice Hall, 1998.

[Gabbay and DeQueiroz 92] D. M. Gabbay and R. De Queiroz, Extending the Curry-Howard interpretation to linear, relevant and other resource logics, *J. Symbolic Logic* 57(4):1319-1365 (1992).

[Gabbay and Kriwaczek 91] D. M. Gabbay and F. Kriwaczek, A family of goal-directed theorem-provers based on conjunction and implication, *Journal of Automated Reasoning*, 7:511-536 (1991).

[Gabbay and Olivetti 97] D. M. Gabbay and N. Olivetti, Algorithmic proof methods and cut elimination for implicational logics - part i, modal implication. *Studia Logica*, 61:237–280, 1998.

[Gabbay and Reyle 84] D. M. Gabbay and U. Reyle, N-Prolog: an Extension of Prolog with Hypothetical Implications. I, *J. Logic Programming* 4:319-355, 1984.

[Gabbay and Reyle 93] D. M. Gabbay and U. Reyle, Computation with run time skolemisation (NProlog part 3), *J. Applied Non Classical Logics* 3(1):93-128 (1993).

[Gabbay 96] D. M. Gabbay, Labelled Deductive Systems (volI), Oxford Logic Guides, Oxford University Press, 1996.

[Gallier 87] S. H. Gallier, Logic for Computer Science, John Wiley, New York, 1987.

[Giordano et al. 92] L. Giordano, A. Martelli and G. F. Rossi. Extending Horn clause logic with implication goals. *Theoretical Computer Science*, 95:43–74, 1992.

[Giordano and Martelli 94] L. Giordano and A. Martelli. Structuring logic programs: a modal approach. *J.of Logic Programming*, 21(2):59–94, 1994.

[Girard 87] J. Y. Girard, Linear logic, *Theoretical Computer Science* 50:1-101 (1987).

[Gore 99] R. GORE Tableaux methods for modal and temporal logics. In M. D'Agostino et al. eds., editor, *Handbook of Tableau Methods.* Kluwer Academic Publisher, 1999.

[Hajek 98] P. HAJEK. **Metamathematics of Fuzzy Logic**, Kluwer Acedemic Publisher, To appear, 1998.

[Halpern and Moses 90] J. Y. HALPERN AND Y. MOSES, Knowledge and common knowledge in a distributed environment, *J. ACM* 37(3):549-587, 1990.

[Harland and Pym 91] J. HARLAND AND D. PYM The uniform proof-theoretic foundation of linear logic programming, *in*Proc. of the 1991 International Logic Programming Symposium, San Diego, October, pp 304-318, 1991.

[Hodas and Miller 91] J. HODAS AND D. MILLER, Logic programming in a fragment of intuitionistic linear logic. *in* (G. Kahn ed.) Sixth Annual Symposium on Logic in Computer Science, Amsterdam, 1991.

[Hosoi 88] T. HOSOI. Gentzen-type Formulation of the Propositional Logic LQ. *Studia Logica*, vol. **47**. pp. 41-48. 1988.

[Hudelmaier 90] J. HUDELMAIER, Decision procedure for propositional n- prolog, *in* (P. Schroeder-Heister ed.) Extensions of Logic Programming, pp. 245-251, Springer Verlag, 1990.

[Kripke 59] S. KRIPKE, The problem of entailment (abstract), *J. Symbolic Logic* 24: 324 (1959).

[Lambek 58] J. LAMBEK The mathematics of sentence structure, *Amer. Math. Monthly* 65: 154-169 (1958).

[Lloyd 84] J. W. LLOYD, *Foundations of Logic Programming*, Springer, Berlin, 1984.

[Loveland 91] D. W. LOVELAND, Near-Horn Prolog and beyond, *Journal of Automated Reasoning*, 7(1):1-26 (1991).

[Loveland 92] D. W. LOVELAND, A comparison of three Prolog extensions, *J. Logic Programming*, 12:25-50 (1992).

[Manna and Pnueli 81] Z. MANNA, A. PNUELI Verification of concurrent programs: the temporal framework. *in* The Correctness problem in Computer Science, Academic Press, London, 1981.

[Martins and Shapiro 88] J. P. MARTINS AND S. C. SHAPIRO, A model for belief revision, *J. Artificial Intelligence*, 35:25-79, 1988.

[Masini92] A. MASINI 2-Sequent calculus: a proof theory of modality, *Annals of Pure and Applied Logics*, 59: 115-149 (1992).

[Massacci94] F. MASSACCI Strongly analytic tableau for normal modal logics. In LNAI 814, pp 723-737, 1994.

[McCarty 88a] L. T. MCCARTY, Clausal Intuitionistic Logic .I. Fixed- Point Semantics. *Journal of Logic Programming*, 5(1): 1–31, 1988.

[McCarty 88b] L.T. MCCARTY, Clausal Intuitionistic Logic .II. Tableau Proof Procedures. *Journal of Logic Programming*, 5(2): 93–132, 1988.

[McRobbie and Belnap 79] M.A. McRobbie, N.D. Belnap, Relevant analytic tableaux, *Studia Logica*, 38: 187–200, 1979.

[Meredith and Prior 64] C. A. Meredith, A. N. Prior Investigations into implicational S5, *Zeitschr f. Logik und Grundlagen d. Math* 10: 203–220, 1964.

[Miglioli et al. 93] P. Miglioli, U. Moscato and M. Ornaghi An improved refutation system for intuitionistic logic, *in* Proceeding of the Second Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Max-Plank-Institut für Informatik, MPI-I-93-213, pp. 169–178, 1993.

[Miller 89] D. A. Miller, A logical analysis of Modules in Logic Programming,*Journal of Logic Programming*, 6: 79–108,1989.

[Miller et al. 91] D. Miller, G. Nadathur, F. Pfenning and A. Scedrov Uniform proofs as a foundation for logic programming, *Annals of Pure and Applied Logic*, 51: 125–157,1991.

[Miller 92] D. A. Miller Abstract Syntax and Logic Programming, *in* Logic Programming: Proc. of the 2nd Russian Conference, Lecture Notes in Artificial Intelligence vol 592, Springer Verlag, pp. 322–337, 1992.

[Nadathur 98] G. Nadathur Uniform provability in classical logic, *Journal of Logic and Computation*, 8(2):209–229,1998.

[Ohlbach 91] H. J. Ohlbach Semantics based translation methods for modal logics, *Journal of Logic and Computation*, 1(5): 691–746, 1991.

[Ohlbach and Wrighston 83] H. J. Ohlbach and G. Wrighston Solving a problem in relevance logic with an automated theorem prover, Technical Report 11, Institut für Informatik, University of Karlsruhe, 1983.

[Pnueli 81] A. Pnueli, The temporal logic of programs, *Theoretical Computer Science*, 13:45–60, 1981.

[Prior 61] A. N. Prior Some axiom-pairs for material and strict implication, *Zeitschr f. Logik und Grundlagen d. Math* 7: 61–65, 1961.

[Prior 63] A. N. Prior The theory of implication, *Zeitschr f. Logik und Grundlagen d. Math* 9: 1–6, 1963.

[Rasiowa 74] H. Rasiowa An Algebraic Approach to Non Classical Logics, North-Holland, 1974.

[Russo 96] A. Russo Generalizing propositional modal logics using labelled deductive systems, *in* (F. Baader, K. Schulz eds.) Frontiers of combining systems, vol. 3 of Applied Logic Series, pp. 57-73, Kluwer Academic Publisher, 1996.

[Sahlqvist 75] ???? Salqvist Completeness and correspondence in first- and second-order semantics of modal logics, *in* (S.Kanger ed.) Proc. of the 3rd Scandinavian Logic Symposium, North-Holland, 1975.

[Seyfried and Huerding 97]

[Thistlewaite et al. 88] P. B. Thistlewaite, M. A. McRobbie and B. K. Meyer Automated Theorem Troving in Non Classical Logics, Pittman, 1988. 1983.

[Turner 85] R. Turner, Logics for Artificial Intelligence, Ellis Horwood Ltd., 1985.

[Wansing 90] H. Wansing Formulas as types for a hierarchy of sublogics of intuitionistic propositional logic, Technical Report n.9/90, Free University of Berlin, 1990.