Labelled Natural Deduction for Substructural Logics

Krysia Broda[*], Marcelo Finger[†], and Alessandra Russo[*]
email: {kb,ar3}@doc.ic.ac.uk, mfinger@ime.usp.br[‡]

November 1997

Technical report DoC. 97/11

Abstract

In this paper a uniform methodology to perform Natural Deduction over the family of linear, relevance and intuitionistic logics is proposed. The methodology follows the Labelled Deductive Systems (LDS) discipline, where the deductive process manipulates *declarative units* – formulas *labelled* according to a *labelling algebra*. In the system described here, labels are either ground terms or variables of a given *labelling language* and inference rules manipulate formulas and labels simultaneously, generating (whenever necessary) constraints on the labels used in the rules. A set of natural deduction style inference rules is given, and the notion of a *derivation* is defined which associates a labelled natural deduction style "structural derivation" with a set of generated constraints. Algorithmic procedures, based on a technique called *resource abduction*, are defined to solve the constraints generated within a derivation, and their termination conditions discussed. A natural deduction derivation is correct with respect to a given substructural logic, if, under the condition that the algorithmic procedures terminate, the associated set of constraints is satisfied with respect to the underlying labelling algebra. This is shown by proving that the natural deduction system is sound and complete with respect to the LKE tableaux system [DG94].

1 Introduction

This paper builds upon the methodology of Labelled Deductive Systems [Gab96] to develop a *uniform* and *abductive* natural deduction system for the family of linear, relevance and intuitionistic logics. The system is uniform in the sense that the set of labelled natural deduction rules is the same for all the three logics under consideration, and abductive in that it incorporates a technique for identifying, when possible, additional assumptions which can be used to detect, from the given derivation, theorems of the given substructural logic.

[*] Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ

[†] Departamento de Ciência da Computação, Instituto de Matemática e Estatística, Universidade de São Paulo

[‡] This work was supported by EPSRC grant GR/J17485

It is widely believed that natural deduction style proof theory is the only formal approach which comes as close as possible to informal rules of reasoning used in everyday discourse[1]. Despite this, in the field of substructural logics little research has been so far devoted to defining proof strategies based on natural deduction, and most work on automated theorem proving has concentrated mainly on sequent calculi [Dôs93]. On the other hand, recent results have shown that Gabbay's methodology based on Labelled Deductive Systems (LDS) [Gab96] provides an ideal framework for developing uniform proof systems for various families of logics. Examples are [DG94] where a uniform labelled semantic tableaux, called LKE system, is defined for a wide family of substructural logics, and [Rus96] in which a uniform natural deduction style proof system, called MLDS, is described for a wide family of propositional and predicate modal logics. Furthermore, in [Gab96] examples of labelled natural deduction rules for some substructural logics have also been described which show some ways of handling labels to allow the same rules to be used in different substructural logics. However, this illustration covers only the implication fragment, no general soundness or completeness results for the rules are given, and moreover it does not include any algorithm for checking relationships between inferred labels. It is mainly given to illustrate a more general claim – LDS can be used to develop a uniform proof system for substructural logics. This paper substantiates this claim for the cases of linear, relevance and intuitionistic logic, also providing some initial results towards the development of automated labelled natural deduction theorem provers for substructural logics.

A uniform labelled natural deduction system for the family of linear, relevance and intuitionistic logics is given, whose set of inference rules is shared by each of these substructural logics. It is well known in the literature [Dôs93] that such logics can be uniquely defined in terms of a set of *operational rules* and a set of *structural rules*. The former are rules associated to each operator, whereas the latter are meta-rules which define how formulae can be used within a derivation. For example, in the case of relevance logic the *permutation* and *contraction* structural rules enforce assumptions to be used *at least* once but not necessarily in the order they are given. Results in the literature (see [Dôs93] for a detailed overview on substructural logics) show that the same set of operational rules is shared by the whole family of substructural logics and that it is each individual set of structural rules which uniquely identifies each logic by defining the properties of the associated consequence relation. Using these rules different theorems can be proved in different substructural logics, even though the basic operational rules are identical. For example, a formula of the form $A \rightarrow ((A \otimes A \rightarrow B) \rightarrow B)$ can be proved to be a theorem of relevance logic but not a theorem of linear logic.

In Section 3 a set of labelled natural deduction rules is given, which is common to the three logics under consideration. These rules are defined on *labelled formulae* and they perform the role of operational rules. Structural rules are instead implemented by different *labelling algebras* (uniquely associated with each individual logic) which define conditions on labels. A derivation is defined as a pair composed of a sequence of inferred labelled formulae, called *structural derivation* and the set (possible empty) of *constraints* on labels generated by the inference rules. The satisfiability of a generated set of constraints depends on the conditions of the underlying labelling algebra. The same set of constraints can be true with respect to one labelling algebra and false with respect to another. It is in this sense that the standard

[1] It is this closeness to the actual reasoning that had prompted Gentzen to put forward his natural deduction approach [Gen35].

manipulation of assumptions given by the use of structural rules (see for instance Gentzen style calculus [Dôs93]) should be seen in this system – to restrict and to control the use and the discharge of assumptions via the use of labels and of a labelling algebra, leaving unchanged the set of labelled natural deduction rules. The three different substructural logics are captured therefore by simply changing the underlying labelling algebra. This facilitates a uniform proof system whose derivation processes are well structured and more human-oriented.

In this paper, attention is restricted to the fragment containing the operators $\rightarrow$ and $\otimes$. Extension to the whole set of substructural operators will be the topic of a future paper. In Section 2 language and syntax of the system are given together with the notion of a *labelling algebra*. The latter is defined in terms of a set of elements, a partial ordering relation and a binary operator. Three different types of labelling algebras are defined by imposing on the binary operator different properties. Results in [DG94, BDR96] have shown that these properties correspond to the structural rules of the substructural consequence relation. In the case of linear logic, the *permutation* property of the consequence relation is captured by requiring the binary operator of the labelling algebras to be *commutative*; for relevance logic, the binary operator satisfies *commutativity* and *contraction*, and for intuitionistic logic it satisfies also *monotonicity*. In Section 3 a set of labelled natural deduction rules is defined and an example is given to illustrate the uniform property of the system. In Section 4 an algorithmic procedure is given which allows the sets of constraints generated within derivations to be solved accordingly with the underlying labelling algebra. The termination property and scope of such algorithms are also discussed. The natural deduction system is shown to be sound and complete with respect to the LKE tableaux system [DG94]. This is proved by considering an extended LKE tableaux system and proving the natural deduction system to be sound and complete with respect to this extended system. This consists in showing that (i) given a natural deduction derivation of a labelled formula with a satisfied set of constraints, there exists a closed LKE refutation of that labelled formula, and that (ii) the converse holds. Part (i) is proved in Section 5.2, whereas part (ii) is briefly discussed in Section 5.1, as already proved in [BDR96]. On the basis of this result, a natural deduction derivation is correct with respect to a given substructural logic if, under the condition that the algorithmic procedure terminates, the set of constraints of the derivation is satisfied with respect to the labelling algebra associated with the underlying logic. The paper concludes with some final discussion and comparisons.

Finally, some remarks regarding syntactic notations. Throughout the paper lower-case letters from u to z are used to refer to terms in the system, whereas upper-case letters denote wffs of the system. Integer subscripts may also be used with any of these letters. Each of the three substructural logics considered in this paper will sometimes be referred to as LL for linear logic, RL for relevance logic and IL for intuitionistic logic.

2 Labelling algebras for substructural logics

In this section basic definitions of the language and syntax of the system are given, together with the notion of a *labelling algebra* and the meaning of a *labelled formula*, for the substructural fragment $\{\rightarrow, \otimes\}$.

The language of the system is defined as an ordered pair $\langle L_{\{\rightarrow,\otimes\}}, L_L \rangle$, where L_L is a *la-*

belling language and $L_{\{\rightarrow,\otimes\}}$ is a standard propositional substructural language restricted to the set of operators $\{\rightarrow,\otimes\}$. The labelling language L_L is composed of the constant symbol 1, a countable set of symbols $\{a,b,\ldots,f,a_1,b_1,\ldots f_1,a_2,b_2,\ldots,f_2,\ldots\}$ called *parameters*, a countable set of variables $\{\gamma,\delta,\gamma_1,\delta_1,\gamma_2,\delta_2,\ldots\}$, a binary function symbol $\circ$ and a binary relation $\sqsubseteq$. Terms of the labelling language are defined inductively as consisting of 1, parameters and variables, together with expressions of the form $x \circ y$ where x and y are terms. Wffs of $L_{\{\rightarrow,\otimes\}}$ are defined in the standard way. Terms are generally referred to as *labels*, the term 1 and the parameters are called *atomic labels*, whereas these and any other term with no variable occurrence are *ground labels*. The syntax of the system is given by two different types of syntactic entity, the *declarative unit* and the *label constraint*. A declarative unit is defined as a pair of the form *formula* : *label*, expressing that a formula is true relative to a piece of information. The formula component is written in $L_{\{\rightarrow,\otimes\}}$ and the label component is a term of L_L. A label constraint in the language L_L is of the form $x \sqsubseteq y$ where x and y are labels.

Both labels and the relation $\sqsubseteq$ are interpreted onto a *labelling algebra*, given in Definition 1. Atomic labels are interpreted in a labelling algebra as themselves in the style of the Herbrand Interpretation. In the rest of this paper the term label will sometimes be used to refer to its interpretation in the algebra. Informally, labels are interpreted as "pieces of information" relative to which formulae are evaluated true or false. The atomic labels given by the parameters of L_L are used to name *particular* pieces of information. The binary relation $\sqsubseteq$ behaves like a kind of Kripke-style accessibility relation between pieces of information, according to which if a piece of information y is accessible from a piece of information x then y verifies all the formulae which are verified at x (if any). For simplicity the same notations are used between the terms of the labelling language and the elements of the labelling algebra, as well as between the binary relation of L_L and the partial ordering of the algebra.

Definition 1 {**Labelling algebra**} A *labelling algebra* $\mathcal{L}_A$ is a tuple$(\mathcal{L},\circ,1,\sqsubseteq)$ such that:

1. $\mathcal{L}$ is a set of *atomic* elements, where $1 \in \mathcal{L}$;
2. $\sqsubseteq$ is a partial ordering
3. $\circ$ is a binary operation on $\mathcal{L}$ which satisfies the following properties:
   (a) associativity: $x \circ (y \circ z) = (x \circ y) \circ z$;
   (b) identity: 1 is the identity element of the operation $\circ$: for every $x \in \mathcal{L}$, $x \circ 1 = 1 \circ x = x$.
   (c) order preserving: for every $x, y \in \mathcal{L}$, if $x \sqsubseteq y$ then $x \circ z \sqsubseteq y \circ z$ and $z \circ x \sqsubseteq z \circ y$ for every $z \in \mathcal{L}$;

The elements of a labelling algebra are pieces of information, or resources, used to verify formulae. The $\circ$ operator allows concatenation of resources. In general, formulae which are verified by means of individual resources are not necessarily verified by the resource composition of the individual ones. The associativity property of the $\circ$ operator means that composed elements of the algebra, which differ only in the way their own components are associated, are identical pieces of information. Adopting the view of a labelling algebra as a "structure" of resources [DG94, BDR96], the associativity property of $\circ$ implies that the composition over the same "sequence" of resources is arbitrary. This implies that if a formula is verified by means of a composition of resources of the form $x \circ (y \circ z)$, then the same formula can be verified composing the same list of resources in a different way.

In addition to the basic properties in Definition 1 the binary operator $\circ$ may satisfy other properties, so defining different types of labelling algebras. In this paper labelling algebras with one or more of the following properties are considered:

$$\begin{array}{ll} \textit{commutativity}: & x \circ y \sqsubseteq y \circ x \\ \textit{contraction}: & x \circ x \sqsubseteq x \\ \textit{monotonicity}: & x \sqsubseteq x \circ y \end{array}$$

The commutativity property states that if a formula is verified by means of a given sequence of resources, than it can be verified changing the order of the resources in the sequence. (Notice that the commutativity property could equally be expressed using the = relation.) The contraction property instead guarantees that if a formula is verified using more than one occurrence of a given resource than it can be verified using a fewer number of occurrences. Finally the monotonicity property gives that if a formula is verified by means of a given resource than it is still verified if this resource is combined with any other resource. Considering each atomic resource as uniquely associated to each formula, a sequence of resources can be read as a sequent of formulae. Under this interpretation, it is possible to see a correspondence between the above three properties of the $\circ$ operator and the properties of the substructural consequence relation. In [Gab96, DG94, BDR96] a proof of such correspondence is given showing that the commutativity property corresponds to the permutation rule, the contraction property to the contraction rule and the monotonicity property to the weakening rule. On the basis of this correspondence it is possible to show that different labelling algebras identify different substructural logics (see [BDR96] for further details). In this paper, three types of labelling algebras are considered. These are respectively the labelling algebra whose operator $\circ$ is commutative, the labelling algebra whose operator $\circ$ satisfies commutativity and contraction, and the labelling algebra whose operator $\circ$ satisfies commutativity, contraction and monotonicity. The first corresponds to Girard's Linear Logic (LL) [Gir87], the second to Relevance Logic (RL) and the third to Intuitionistic Logic. This is summarised in Table 1, where the notation $\mathcal{L}_\Delta$ with $\Delta \in \{\text{LL}, \text{RL}, \text{IL}\}$ is also introduced. In addition, the notation $\mathcal{L}^S_\Delta$ is used to denote the particular algebra $\mathcal{L}_\Delta$ whose set of elements $\mathcal{L}$ is equal to a given set S.

The meaning of a declarative unit is defined in terms of a *valuation* function.

Definition 2 {**Valuation function**} Let $\mathcal{L}_A = (\mathcal{L}, \circ, 1, \sqsubseteq)$ be a labelling algebra and let $\mathbf{F}$ be the set of wffs of $L_{\{\rightarrow,\otimes\}}$. A *valuation* over $\mathcal{L}_A$ is a mapping $V : \mathbf{F} \times \mathcal{L} \longrightarrow \{T, F\}$ satisfying the following conditions:

1. For all formulae A, if $V(A, x) = T$ and $x \sqsubseteq y$, then $V(A, y) = T$.

2. For all formulae A, if $V(A, x) = T$ for some $x \in \mathcal{L}$, then there exists also $a \in \mathcal{L}$, called the A-*characteristic*, where $V(A, a) = T$ and a is the *least* such element with respect to the ordering $\sqsubseteq$. That is, if, for any $y \in \mathcal{L}$, $V(A, y) = T$ then $a \sqsubseteq y$.

3. For each wff of the form $A \rightarrow B$ and for each label x
$V(A \rightarrow B, x) = T \Leftrightarrow \forall y[V(A, y) = T \text{ implies } V(B, x \circ y) = T]$.

Conditions on $\sqsubseteq$	Labelling algebras
$\{x \circ y \sqsubseteq y \circ x\}$	$\mathcal{L}_{\mathrm{LL}}$
$\{x \circ y \sqsubseteq y \circ x,$ $x \circ x \sqsubseteq x\}$	$\mathcal{L}_{\mathrm{RL}}$
$\{x \circ y \sqsubseteq y \circ x,$ $x \circ x \sqsubseteq x,$ $x \sqsubseteq x \circ y\}$	$\mathcal{L}_{\mathrm{IL}}$

Table 1: Classes of labelling algebras

4. For each wff of the form $A \otimes B$ and for each label x
   $V(A \otimes B, x) = T \Leftrightarrow \exists y, z[y \circ z \sqsubseteq x$ and $V(A, y) = T$ and $V(B, z) = T]$.

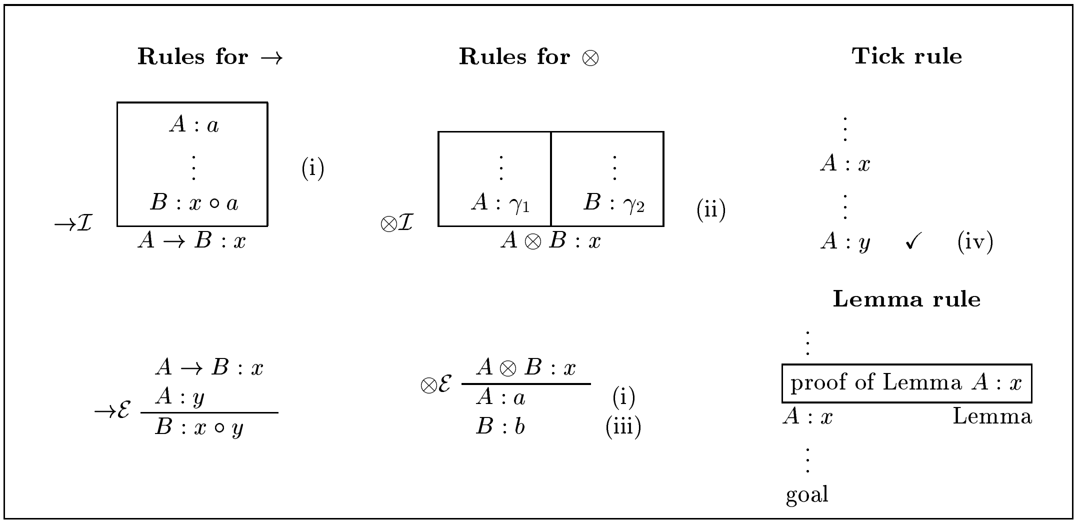
Condition (1) in the above definition expresses the "hereditary" property of the truth values with respect to the partial ordering $\sqsubseteq$, condition (2) plays an important role in the definition of some of the natural deduction rules, and in proving the correspondence of the natural deduction system with respect to the LKE system, whereas conditions (3) and (4) provide the semantic meaning of the two substructural operators $\rightarrow$ and $\otimes$ respectively. Using the above notion of a valuation function, a declarative unit $A : x$ is said to be *satisfied* if and only if $V(A, x) = T$, where the argument x of V is the interpretation of the label x in the labelling algebra. Given a labelling algebra $\mathcal{L}_\Delta$, for some $\Delta \in \{$LL, RL, IL$\}$, and a valuation V over $\mathcal{L}_\Delta$, the tuple $\langle \mathcal{L}_A, V \rangle$ can be seen as a *semantic structure* for the fragment $\{\rightarrow, \otimes\}$ of the substructural logic Δ.

3 A Uniform Natural Deduction System

In this section a uniform labelled natural deduction style proof system is described for the fragment $\{\rightarrow, \otimes\}$ of linear, relevance and intuitionistic logics. The set of inference rules is given together with the definitions of a "structural derivation" and "label constraints". An example is also given which shows how the same set of inference rules can be used to construct derivations in any of the three substructural logics.

It is strongly believed that this system expands the study of the natural deduction formalism, initially developed by Gabbay in [Gab96], showing how the naturalness and the closeness to actual reasoning typical of natural deduction calculus can also capture substructural deductive processes. There is in fact no reason for restricting the attention only to tableau methods and sequent calculi [DG94, Dôs93]. Natural deduction systems can be equally expressive.

A formal definition of the inference rules is given in Table 2, adopting a presentation style first introduced in [BEKV94] for the definition of a classical natural deduction style proof system. A *solo parameter* is an atomic label such that any other atomic occurrences within a structural derivation labels only the formula it is first introduced with, and which may occur (as atomic



(i) a is a *solo* parameter
(ii) $\gamma_1 \circ \gamma_2 \sqsubseteq x$
(iii) b is a *solo* parameter, and $a \circ b \sqsubseteq x$
(iv) $x \sqsubseteq y$

Table 2: ND Rules for the substructural fragment $\{\rightarrow, \otimes\}$.

label) repeatedly in the derivation whenever such a formula has to be re-introduced. In the cases of $\rightarrow\mathcal{I}$ and $\otimes\mathcal{E}$, the parameters a and b are respectively solo parameters. The two introduction rules can be interpreted procedurally as follows: "to show $A \rightarrow B : x$, assume $A : a$ and show $B : x \circ a$", where a is a solo parameter of L_L (i.e. it does not appear as an atomic label of any formula other than A, and any re-introduction of A is labelled with a), and "to show $A \otimes B : x$ show $A : \gamma_1$ and $B : \gamma_2$", where $\gamma_1 \circ \gamma_2 \sqsubseteq x$. They are both used in reasoning backwards (i.e. reasoning from the *goal formula* needed to be proven), whereas the two elimination rules are used forward (i.e. reasoning from the given assumptions).

The introduction and the elimination rules for the $\rightarrow$ operator together reflect the semantic interpretation of $\rightarrow$ given in Section 2 (see condition (3) of Definition 2).

As for the operator $\otimes$, the introduction and the elimination rules, validated by their *satisfied* side conditions, together reflect the semantic interpretation of $\otimes$ given in Definition 2 (see condition (4)). Specifically, the side condition of the $\otimes\mathcal{I}$ rule corresponds to the condition on the labels given in the right-hand side of the equivalence (4) where the solo parameters used in the $\otimes\mathcal{E}$ rule can be seen as "Skolem" constants.

The "Tick" rule allows complete sub-proofs to be recognised. It enables forwards and backwards reasoning within a derivation to be "linked" together. For example, the application of the $\otimes\mathcal{I}$ rule is conditioned to the provability of its respective sub formulae (or *sub-goals*). These sub-goals may be proved by inferring in their respective sub-derivations, with some forward reasoning steps the same formulae with labels equal or smaller than the one appearing in the sub-goals. Applications of the tick rule would allow in this case the "closure" of the two sub-derivations. From a semantic point of view, this rule, together with its *satisfied* side condition, reflect the hereditary property of the valuation function (i.e. condition (1) of

Definition 2).

The "Lemma" rule can be seen as a way of incorporating the notion of "cut" into the natural deduction system. It is used in a derivation whenever the operational rules cannot derive any new formula from a given set of assumptions. The rule provides a way of introducing a new relevant formula not as an assumption but as a lemma whose appropriate proof is constructed in the sub-derivation of the rule itself. An example of a derivation that requires this rule is that of proving $C : c$ from the assumption $(A \rightarrow A) \rightarrow C : c$ – the lemma rule is needed because of the formulation of the $\rightarrow\mathcal{E}$ rule. The introduction of such a rule facilitates also an easier proof of correspondence with the LKE system which includes the cut rule explicitly. An example derivation which uses the lemma rule is given in Figure 1.

In the $\otimes\mathcal{I}$ and Lemma rules, boxes mainly have the function of separating sub-derivations. It is only in the case of $\rightarrow\mathcal{I}$ that the introduction of a box implies also the introduction of a new assumption and that its closure implies the discharge of the same assumption. This standard way of using boxes to handle additional assumptions is in this system redundant, labels could provide a sufficient book-keeping device on their own. However, the box presentation technique is here preserved, as it helps in *structuring* the derivations. It is this structural property of the derivations that facilitates an easy search of solutions for the generated set of constraints. This is one of the main advantages of this system which can make it preferable to other proof systems. For example in the LKE tableau system [DG94], there is no apparent structure. This can make sometimes the underlying reasoning hard to follow as well as the resulting constraints difficult to solve.

Label constraints. Most of the labelled natural deduction rules given in Table 2 have side conditions. These, except condition (i) and the first part of condition (iii), are called *label constraints*, and are of the form $x \sqsubseteq y$ where x and y are labels in the language L_L. For any given constraint $x \sqsubseteq y$, the terms x and y will often be referred to as the left-hand-side (LHS) and the right-hand-side (RHS) of the constraint. The satisfiability of label constraints depends on the properties of the underlying labelling algebra. In Section 4 a basic algorithmic procedure is defined for solving such label constraints and extensions of this procedure are also given which take into account the specific additional properties of each type of labelling algebra. The process of detecting whether a derivation is correct with respect to a given logic reduces to resolving the set of label constraints generated within the derivation. The algorithms use also a technique called *resource abduction* which, whenever they terminate, provide a way of "abducing" the additional assumptions which can be used to detect theorems of a given substructural logic.

The label constraints generated within a natural deduction derivation are of two different types: *imposed constraints* (*IC*s) and *required constraints* (*RC*s). The imposed constraints are introduced by the application of the $\otimes\mathcal{E}$ rule (see the second part of condition (iii) shown in Table 2) on $\otimes$-formulae. Their validity is "imposed" by the occurrences of the $\otimes$-formulae, consistently with the semantic interpretation of the $\otimes$ operator. Notice that, in this type of constraint, variables occur only on the right hand side. The required constraints are instead generated by the $\otimes\mathcal{I}$ rule and by the Tick rule (respectively conditions (ii) and (iv) shown in Table 2).

The notion of a derivation. In this system the notion of a derivation extends the standard notion of a natural deduction derivation. A derivation is a pair composed of a sequence of inferred declarative units, called *structural derivation*, and a set of generated constraints. A structural derivation does not guarantee itself the derivability of a formula in a given substructural logic. Arbitrary structural derivations could in fact be constructed, but only those whose associated set of generated label constraints is satisfied in the underlying labelling algebra, are *correct* derivations. This is captured by the following definitions.

Definition 3 $\{$**Set of initial assumptions**$\}$ A set of initial assumptions T is a set of declarative units of the form $F : f$, where F is a wff of $L_{\{\rightarrow,\otimes\}}$ and f is a solo parameter that does not appear as an atomic label of any other initial assumption in T.

Definition 4 $\{$**Structural derivation**$\}$ Let A be a wff written in the language $\mathcal{L}_{\{\rightarrow,\otimes\}}$, let x be a label and let T be a (possibly empty) set of initial assumptions. A *structural derivation* of the declarative unit $A : x$ from T is a finite sequence of pairs $\langle\alpha_1, r_1\rangle$, $\langle\alpha_2, r_2\rangle$, ..., $\langle\alpha_k, r_k\rangle$, such that $\alpha_k = A : x$, and for each $1 \leq i \leq k$, if α_i is an assumption then r_i is empty, otherwise α_i is the declarative unit inferred by the application of the labelled natural deduction rule r_i. The set of atomic labels occurring in this sequence of pairs, extended with the label 1, is called the *domain* of the structural derivation.

Definition 5 $\{$**Satisfied label constraints and solutions**$\}$ Let $\Delta \in \{$LL, RL, IL$\}$, let $IC_1, \ldots, IC_n$ and $RC_1, \ldots, RC_m$, where $n \geq 0$ and $m \geq 0$ be a set of imposed and required constraints such that $\gamma_1, \ldots, \gamma_k$, with $k \geq 0$, are the variables that appear in the constraints and the set of atomic labels occurring in the constraints be D. Let $\mathcal{L}^D_\Delta$ be the labelling algebra associated with D. Then the set of constraints $\{IC_1, \ldots, IC_n, RC_1, \ldots, RC_m\}$ is *satisfied in* $\mathcal{L}^D_\Delta$ iff there exists a ground instantiation in $\mathcal{L}^D_\Delta$ of $\gamma_1, \ldots, \gamma_k$, called a *solution*, such that $RC_1 \wedge \ldots \wedge RC_m$ is true in $\mathcal{L}^D_\Delta$ extended with the conjunction $IC_1 \wedge \ldots \wedge IC_n$ of imposed constraints.

In Section 4 algorithmic procedures for "solving" a set of label constraints are described.

Definition 6 $\{$**Derivability**$\}$ Given a $\Delta \in \{LL, RL, IL\}$, a declarative unit $A : x$ and a (possibly empty) set T of initial assumptions, $A : x$ is *derivable* from T in the substructural logic Δ, written $T \vdash_\Delta A : x$, if there exists a tuple $\langle\Gamma, \{IC_1, \ldots, IC_n, RC_1, \ldots, RC_m\}\rangle$, where $\Gamma = \{\Gamma_1, \ldots, \Gamma_k\}$ is a structural derivation of $A : x$ from T with domain D and, for each $1 \leq i \leq n$ and $1 \leq j \leq m$, IC_i and RC_j are respectively the imposed and the required constraints generated by the inference rules in the structural derivation, and the set $\{IC_1, \ldots, IC_n, RC_1, \ldots, RC_m\}$ is satisfied in the labelling algebra $\mathcal{L}^D_\Delta$. The tuple $\langle\{\Gamma_1, \ldots, \Gamma_k\}, \{IC_1, \ldots, IC_n, RC_1, \ldots, RC_m\}\rangle$ is called a *derivation* in Δ of $A : x$ from T.

Theorems are formulae proved to be derivable from an empty set of initial assumptions, with the atomic label 1. This is formally defined as follows.

Definition 7 $\{$**Theoremhood**$\}$ Let $\Delta \in \{$LL, RL, IL$\}$, then a formula A is a *theorem* of Δ if the declarative unit $A : 1$ is derivable in Δ from an empty theory. This is sometimes written as $\emptyset \vdash_\Delta A : 1$, or simply $\vdash_\Delta A : 1$.

3.1 Uniformly Coping with Different Logics (An example)

The uniform property of the labelled natural deduction rules claimed in the previous section is here illustrated with an example. This consists of taking a formula, known to be a theorem of relevance logic but not of linear logic, constructing a structural derivation, which is the same whatever underlying logic, and then describing how different logics are accommodated within the system by means of appropriate solving processes on the label constraints.

The formula under consideration is $A \to ((A \otimes A \to B) \to B)$. By Definition 7, to show that this formula is a theorem of LL or of RL, it is necessary to show that there exists a derivation with domain D of the declarative unit $A : 1$ whose set of label constraints is satisfied in $\mathcal{L}^D_{LL}$ or $\mathcal{L}^D_{RL}$. The case of LL is considered first. A structural derivation is given in Figure 1, whose domain $D = \{a, b, 1\}$. The set of generated imposed constraints is empty, since no application of the $\otimes\mathcal{E}$ rule is made, whereas the set of generated required constraints is given in (1).

$$\{a \sqsubseteq \gamma_1,\ \ a \sqsubseteq \gamma_2,\ \ \gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3,\ \ b \circ \gamma_3 \sqsubseteq 1 \circ a \circ b\} \tag{1}$$

1	$A : a$			
2	$A \otimes A \to B : b$			
3	$A : \gamma_1$	$\surd\ \ a \sqsubseteq \gamma_1$	$A : \gamma_2$	$\surd\ \ a \sqsubseteq \gamma_2$
4	$A \otimes A : \gamma_3$			$\otimes\mathcal{I}\ \gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3$
5	$A \otimes A : \gamma_3$			lemma
6	$B : b \circ \gamma_3$			$\to\mathcal{E}$
7	$B : 1 \circ a \circ b$			$\surd\ b \circ \gamma_3 \sqsubseteq 1 \circ a \circ b$
8	$(A \otimes A \to B) \to B : 1 \circ a$			$\to\mathcal{I}$
9	$A \to ((A \otimes A \to B) \to B) : 1$			$\to\mathcal{I}$

Figure 1: An Example of structural derivation.

This derivation is a correct derivation in LL if and only if its associated set of constraints is satisfied in $\mathcal{L}^D_{LL}$. In general, a set of constraints is satisfied if each required constraint "succeeds" for some instantiation values of the variables occurring in x and in y, with respect to the properties of the underlying labelling algebras extended with the instantiated imposed constraints. These values are called *solutions* of the constraints and the process of searching for solutions of a given required constraint is called the *solving process*. In the case of LL, the associated type of labelling algebras include the commutativity property of $\circ$. In solving a required constraint, the two properties of associativity and commutativity need be taken into account. This means, for example, allow required constraint of the form $a \circ b \sqsubseteq \gamma \circ a$ to succeed for $\gamma = b$, as by commutativity it could be reduced to the constraint $b \circ a \sqsubseteq \gamma \circ a$. In the above example derivation, the required constraints $a \sqsubseteq \gamma_1$ and $a \sqsubseteq \gamma_2$ are satisfied in $\mathcal{L}^D_{LL}$ only if $\gamma_1 = a$ and $\gamma_2 = a$ respectively, and the constraint $\gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3$ is satisfied only if γ_3 is instantiated to $a \circ a$. The last constraint $b \circ \gamma_3 \sqsubseteq 1 \circ a \circ b$ is thus reduced to $b \circ a \circ a \sqsubseteq 1 \circ a \circ b$, which, by the identity property of the element 1, is equivalent to $b \circ a \circ a \sqsubseteq a \circ b$, which is not true in $\mathcal{L}_{LL}^{D}$. The set of required constraints associated with the structural derivation given in Figure 1 is therefore not satisfied in $\mathcal{L}_{\mathrm{LL}}^{D}$ and the proof is not a derivation in LL.

The case of Relevance Logic is now considered. To show that the same formula is a theorem of Relevance Logic it is necessary to show that there exists a derivation of the declarative unit $A \to ((A \otimes A \to B) \to B) : 1$. This means to show that there exists a structural derivation with domain D whose set of constraints is satisfied in $\mathcal{L}_{RL}^{D}$. The same structural derivation, with domain D, given in Figure 1 is generated also in this case – rule applications do not depend on the particular underlying logic. However, to show that it is a derivation of RL it is necessary to show that the set of label constraints in (1) is satisfied with respect to $\mathcal{L}_{RL}^{D}$. This labelling algebra includes both the commutativity and contraction properties of $\circ$. The contraction property allows the left-hand-side of a constraint to contain more occurrences than the right-hand-side of any label appearing in the right-hand-side. So, for example, a required constraint of the form $a \circ b \circ b \sqsubseteq a \circ b$, which would not succeed in $\mathcal{L}_{LL}^{D}$, does instead succeed in $\mathcal{L}_{\mathrm{RL}}^{D}$.

To retain the solution process of a constraint $x \sqsubseteq y$ and allow also for contraction, a way of "evening up" the atomic occurrences in the two parts is needed. This is done via the use of *slack variables*. For each required constraint $x \sqsubseteq y$, an additional variable, denoted with δ and called a *slack variable*, is added to its right-hand-side, giving the new constraint $x \sqsubseteq y \circ \delta$. This variable can *only* be unified with the *atomic labels contracted* in the left-hand-side. If no contraction occurs on the left-hand-side of the required constraint then the slack variable is unified with 1. When instantiations of all the variables (slack and non) occurring in a constraint are found, which satisfy the equality $x = y \circ \delta$, it is necessary to check that the value of the slack variable indeed corresponds to contracted labels. If this checking fails than the instantiations found are rejected. This is shown below where the constraints in (1) are solved with respect to $\mathcal{L}_{RL}^{D}$.

To check whether the set (1) of required constraints is satisfied in $\mathcal{L}_{RL}^{D}$, it is first necessary to add slack variables to their right-hand-sides. The set then becomes $\{a \sqsubseteq \gamma_1 \circ \delta_1,\ \ a \sqsubseteq \gamma_2 \circ \delta_2,\ \ \gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3 \circ \delta_3,\ \ b \circ \gamma_3 \sqsubseteq \gamma \circ a \circ b \circ \delta_4\}$. The first two constraints are satisfied by the instantiation $\gamma_1 = \gamma_2 = a$ and $\delta_1 = \delta_2 = 1$. (No contraction takes place here.) The third inequality becomes under this instantiation $a \circ a \sqsubseteq \gamma_3 \circ \delta_3$. In this case γ_3 can be either a or $a \circ a$ or 1. The first instantiation would make δ_3 equal to a, which means that a contraction of one occurrence of a has occurred in the left-hand-side of the constraint. The second instantiation would make δ_3 equal to 1, which means no contraction has occurred. The third instantiation ($\gamma_3 = 1$) would instead make $\delta_3 = a \circ a$, which means that all the occurrences of a in the left-hand-side are included in δ_3. The last of these instantiations – $\{\gamma_3 = 1, \delta_3 = a \circ a\}$ – is however rejected as a slack variable can only "absorb" contracted occurrences. Substituting either of the other two instantiations (i.e. $\{\gamma_3 = a, \delta_3 = a\}$ and $\{\gamma_3 = a \circ a, \delta_3 = 1\}$) to the fourth inequality, the solution $\gamma = 1$ can be obtained, thus showing that $A \to ((A \otimes A \to B) \to B) : \gamma$ is a theorem of RL.

This technique of using slack variables on the right-hand-side of a required constraint is still applicable in Intuitionistic Logic, as contraction is also a property of the labelling algebras associated with this logic. However, for these type of labelling algebras $\mathcal{L}_{\mathrm{IL}}$ the solving process has also to take into account the monotonicity property. Monotonicity means that constraints with right-hand-side bigger than the left-hand-side succeed. Thus to use the solution process illustrated above, additional slack variables need also to be added to the left-hand-side of a required constraint to even up the additional labels occurring in its right-hand-side. A full detailed description of the solving process is given in Section 4.2.

From the example of constraints solution given above, it is evident that the order in which constraints of a given set are solved facilitates their solutions. Solving a constraint of the form $\gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3$ without knowing any particular instantiation value for any of the three occurring variables is in fact much harder. In Section 4.1 it is shown that for any given set of generated required constraints it is always possible to define an ordering (similar to the one implicitly used here) which facilitates the search for solutions.

3.2 Resource Abduction

So far it has been illustrated, via an example, how the same structural derivation can be constructed in different logics and how different logics can be accommodated within the same basic solving process for label constraints. An extension, and more general approach, to this proof system is given by the use of a technique called *resource abduction*. This consists of showing that for a given formula A there exists a derivation of the declarative unit $A : \gamma$ (for an arbitrary variable γ) instead of the declarative unit $A : 1$. In so doing, some of the constraints generated within the derivation would refer to the variable γ and their solutions would include an instantiation value for γ. If $\gamma = 1$ is a solution then the given formula is a theorem of the underlying logic. If this is not the case then a solution for γ provides information of the atomic formulae that could be added to the given formula (as initial assumptions) in order to prove this new formula is a theorem of the underlying logic. Consider this technique in the above example. The declarative unit to be proven is $A \rightarrow ((A \otimes A \rightarrow B) \rightarrow B) : \gamma$. A structural derivation identical to that given in Figure 1 is constructed, but with each occurrence of 1 in the labels replaced by γ. The associated set of required constraints is given in (2).

$$\{a \sqsubseteq \gamma_1,\ \ a \sqsubseteq \gamma_2,\ \ \gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3,\ \ b \circ \gamma_3 \sqsubseteq \gamma \circ a \circ b\} \qquad (2)$$

In the case of LL, the only solution of the constraints in (2) is $\gamma_1 = a$, $\gamma_2 = a$, $\gamma_3 = a \circ a$ and $\gamma = a$. Since the value for γ is not 1, the structural derivation in Figure 1 is not a derivation in LL, as already concluded above. However, the value $\gamma = a$ together with the fact that a is a solo parameter occurring in the derivation as atomic label of A, indicates that adding a "missing A assumption" to the initial formula will prove the new formula to be a theorem of LL. Introducing such a "missing resource" in the form of "$A \rightarrow$" at the front of the initial formula would give the new formula $A \rightarrow (A \rightarrow ((A \otimes A \rightarrow B) \rightarrow B))$. To prove that this is a theorem of LL, a structural derivation is first constructed as shown in Figure 2, whose associated set of constraints is given in (3).

$$\{a \sqsubseteq \gamma_1,\ \ a \sqsubseteq \gamma_2,\ \ \gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3,\ \ b \circ \gamma_3 \sqsubseteq \gamma \circ a \circ a \circ b\} \qquad (3)$$

A solution of (3) in $\mathcal{L}^D_{LL}$ is $\gamma_1 = \gamma_2 = a$, $\gamma_3 = a \circ a$ and $\gamma = 1$. The existence of the value $\gamma = 1$ shows that the new formula is a theorem of LL. The use of such a resource abduction technique does not affect derivations which are already correct with respect to a given logic, without resource abduction. For instance, using this technique in the case of RL, the constraints in (2), rewritten so to include slack variables, can still be solved in the same way as shown for the set (1), giving $\gamma = 1$ and so proving that the given formula is a theorem of RL. No additional resource is in this case necessary as the initial formula is already a theorem of RL.

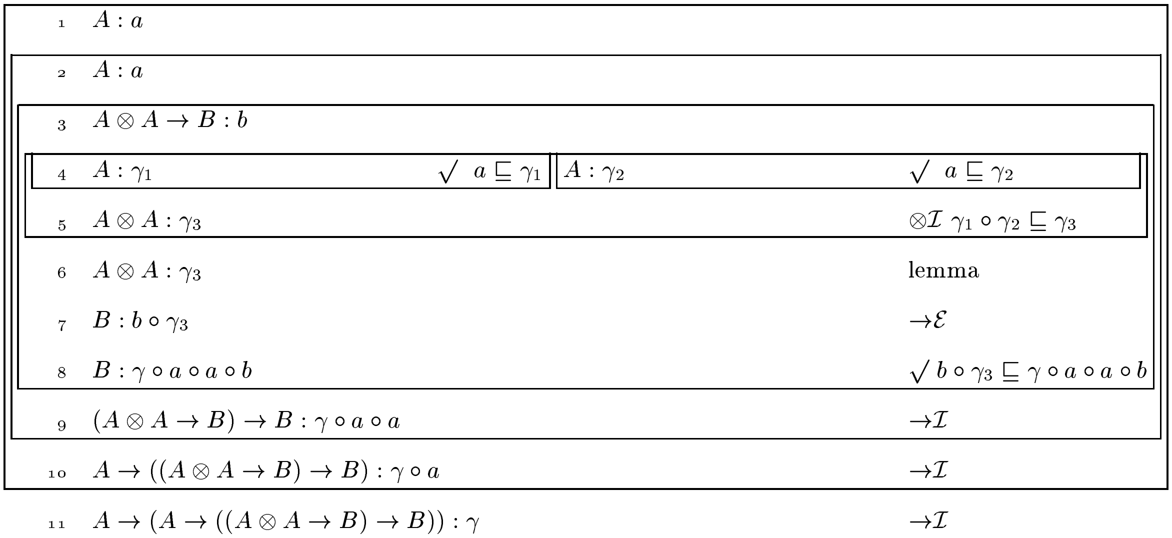

1 $A : a$
2 $A : a$
3 $A \otimes A \to B : b$
4 $A : \gamma_1$ $\surd\ a \sqsubseteq \gamma_1$ $A : \gamma_2$ $\surd\ a \sqsubseteq \gamma_2$
5 $A \otimes A : \gamma_3$ $\otimes\mathcal{I}\ \gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3$
6 $A \otimes A : \gamma_3$ lemma
7 $B : b \circ \gamma_3$ $\to\mathcal{E}$
8 $B : \gamma \circ a \circ a \circ b$ $\surd\ b \circ \gamma_3 \sqsubseteq \gamma \circ a \circ a \circ b$
9 $(A \otimes A \to B) \to B : \gamma \circ a \circ a$ $\to\mathcal{I}$
10 $A \to ((A \otimes A \to B) \to B) : \gamma \circ a$ $\to\mathcal{I}$
11 $A \to (A \to ((A \otimes A \to B) \to B)) : \gamma$ $\to\mathcal{I}$

Figure 2: Structural derivation with resource abduction.

This is a very nice illustration of the power of the resource abduction technique used in this proof system. Structural derivations which result to be not derivations of a given logic may be used to "abduce" other theorems of that logic. To the best of the authors' knowledge, there is not theorem prover in the literature with such a characteristic.

With respect to a more general idea of abduction, the above technique could also be used to identify the declarative units necessary to be added to a (possibly empty) set T of initial assumptions in order to allow the derivation of a given declarative unit from T. A brief example is given here. A structural derivation with domain $D = \{a, b, 1\}$ is given in Figure 3, whose associated set of constraints is given in (4). Notice that in this structural derivation

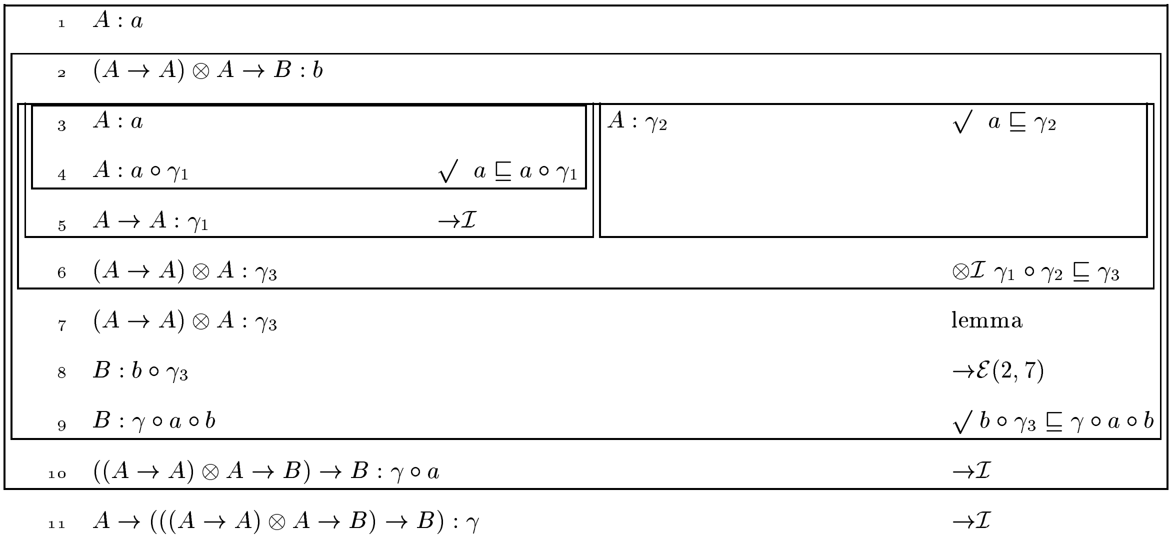

1 $A : a$
2 $(A \to A) \otimes A \to B : b$
3 $A : a$
4 $A : a \circ \gamma_1$ $\surd\ a \sqsubseteq a \circ \gamma_1$
5 $A \to A : \gamma_1$ $\to\mathcal{I}$
$A : \gamma_2$ $\surd\ a \sqsubseteq \gamma_2$
6 $(A \to A) \otimes A : \gamma_3$ $\otimes\mathcal{I}\ \gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3$
7 $(A \to A) \otimes A : \gamma_3$ lemma
8 $B : b \circ \gamma_3$ $\to\mathcal{E}(2,7)$
9 $B : \gamma \circ a \circ b$ $\surd\ b \circ \gamma_3 \sqsubseteq \gamma \circ a \circ b$
10 $((A \to A) \otimes A \to B) \to B : \gamma \circ a$ $\to\mathcal{I}$
11 $A \to (((A \to A) \otimes A \to B) \to B) : \gamma$ $\to\mathcal{I}$

Figure 3: Another structural derivation with resource abduction.

the declarative unit $A : a$ at line 3 is not necessary, as already assumed in the external box at line 1.

$$\{a \sqsubseteq a \circ \gamma_1,\ a \sqsubseteq \gamma_2,\ \gamma_1 \circ \gamma_2 \sqsubseteq \gamma_3,\ b \circ \gamma_3 \sqsubseteq \gamma \circ a \circ b\} \tag{4}$$

The constraints in (4) are solved in $\mathcal{L}^D_{LL}$ considering $\gamma_3 = a$ and $\gamma = 1$. Suppose now that a derivation from an empty initial set T of assumptions of the declarative unit $(((A \rightarrow A) \otimes A) \rightarrow B) \rightarrow B : \gamma$ is required. From the example given in Figure 3, the same structural derivation would be expected but with the solution $\gamma = a$ in its associated required constraints. However, this is not necessary the case. In fact no structural derivation can be constructed for $(((A \rightarrow A) \otimes A) \rightarrow B) \rightarrow B : \gamma$, considering an empty set T of initial assumptions, as the assumption at line 1 is no longer present, and so $A : \gamma_2$ at line 3 cannot be proved. What is needed is the local assumption at line 3 in Figure 3 to be included in T as an initial global assumption. The declarative unit $A : a$ can then be abduced as part of the initial theory T. Hence in order to find a derivation of a given declarative unit, additional declarative units need sometimes to be added to the initial theory T. These are always of the form $F : f$, where F is a subformula of the given formula and f is a solo parameter that does not appear as an atomic label of any formula other than F.

4 Solving Constraints

In this section formal definitions of the algorithmic procedures adopted to solve label constraints are given for each considered substructural logic. As mentioned in the previous section, the case of LL is the simplest one. Only the properties of associativity and commutativity of the operator $\circ$ need to be taken into account when defining its algorithmic procedure. For the cases of RL and IL, extensions of the LL's algorithmic procedure are defined which accommodate the additional properties of contraction and monotonicity.

Before going into the details of the algorithmic procedures it is important to briefly discuss the issue of defining such algorithms. It has been stated in the previous sections that within this system the same structural derivation with domain D can be constructed in different substructural logics, as the "decision" process whether the derivation is a derivation in Δ is left to the solving process of the label constraints generated in the derivation. To fully satisfy such a decision requirement, the solving process should be able to identify when the given formula is a theorem of Δ or not. Because of the resource abduction technique, this means providing an algorithm which solves (whenever possible) every required constraint associated with the structural derivation and which terminates giving a possible instantiation (i.e. solution) for the variables in the constraints. When the algorithm terminates with a solution which satisfies the required constraints in the $\mathcal{L}^D_\Delta$ extended with the instantiated imposed constraints, the derivation is a derivation in Δ.

Given a derivation Π of a declarative unit $A : x$ from a set of initial assumptions T, with the associated domain D and set S of constraints, the search of solutions for S is made by the algorithm within the domain D. By Definition 5, the set S of constraints is satisfied if there exists an instantiation for its variables in $\mathcal{L}^D_\Delta$. The solving process is able to find such instantiation as long as it is allowed for the number of steps to be unlimited. The declarative unit $A : x$ is then proved to be derivable in Δ from T. The solving process is composed of two steps, namely *instantiation* and *expansion* steps. The first one instantiates the variables occurring in a RC whereas the expansion step allows the generation of a new constraint from a given required constraint, using the information given by the imposed constraints. These two steps are described in detail in Sections 4.2 and 4.3 respectively. Because of the expansion steps, the termination property of this process is not always guaranteed. Examples can be in fact constructed in which infinitely many instantiations of the variables occurring in the RCs can be found using the ICs and the properties of the underlying labelling algebra, which would still not satisfy all the label constraints. (See example illustrated in Figure 12.) In such cases the solving process would not terminate and no kind of decision (neither positive nor negative) could be reached. The process is therefore semi-decidable. To control the search for solutions an incremental limit on the number of times ICs are used to generate new possible solutions needs to be imposed.

The following theorem captures this discussion. Its proof is justified by the formal definitions of the algorithmic procedures and of the solving process given throughout the rest of this section.

Theorem 1 {**Partial Correctness**} *Let $\Delta \in \{LL, RL, IL\}$, $E : \gamma$ be a declarative unit and T a set of initial assumptions. Let $\langle \Gamma, \{IC_1, \ldots, IC_n RC_1, \ldots, RC_m\}\rangle$ be a derivation of $E : \gamma$ from T with domain D. If a finite restriction is imposed on the number of expansion steps, then the solving process terminates. If the set $\{IC_1, \ldots, IC_n, RC_1, \ldots, RC_m\}$ of constraints is satisfied in $\mathcal{L}^D_\Delta$, then there exists such a finite restriction on the number of expansion steps for which the solving process terminates giving the solutions. If the set $\{IC_1, \ldots, IC_n RC_1, \ldots, RC_m\}$ is not satisfied in $\mathcal{L}^D_\Delta$, there there is no finite number of expansion steps in which the solving process terminates giving a solution.*

As a consequence of Theorem 1, whenever $T = \emptyset$ if the set of constraints is satisfied and the variable γ in the declarative unit $E : \gamma$ is instantiated to 1, then the formula E is said to be a theorem of Δ. If γ can only be instantiated with values different from 1, then E has not been shown to be a theorem of the underlying logic Δ as there may be a different structural derivation of $E : 1$, but the formula of the form $T_1 \rightarrow (T_2 \rightarrow (\ldots T_k \rightarrow E) \ldots)$, where T_i is the subformula associated with the i-th element in the instantiation of γ, is a theorem in Δ.

Before the solving process is applied on a set of generated label constraints, the set of required constraints is *ordered* to facilitate an easier search for solutions. This ordering procedure is defined first in the following section and the definition of the solving process is then given in Section 4.4.

4.1 Requirements for Required Constraints

This section shows that required constraints generated within a derivation can be brought to a simple format, which facilitates their solutions. Firstly, some remarks about notation. In what follows labels of the form $a_1 \circ a_2 \circ \ldots \circ a_n$ are written more simply as $a_1 a_2 \ldots a_n$. Since $\circ$ is associative and commutative $a_1 a_2 \ldots a_n$ can be interpreted as a multi-set composed of the elements a_1, a_2, ..., a_n. Therefore standard operations on multisets can be used on such terms. Specifically, $a_1 a_2 \ldots a_n - b_1 b_2 \ldots b_m$, denotes the label obtained by applying the multi-set "subtraction" operation on the two labels $a_1 a_2 \ldots a_n$ and $b_1 b_2 \ldots b_m$. Analogously $x \in a_1 a_2 \ldots a_n$, denotes a single occurrence of x in the multiset $a_1 a_2 \ldots a_n$. Similarly for $\subseteq$.

Moreover, throughout the rest of this paper the notation $v_L(RC_i)$ and $v_R(RC_i)$ is used to refer to the set of variables occurring on the LHS and to the set of variables occurring on the RHS of a given required constraint RC_i respectively.

As mentioned in the previous sections a required constraint may contain variables in both its left and right hand sides. Its satisfiability consists in finding instantiations for each of these variables. In this section it is shown that a set of RCs generated in a natural deduction derivation satisfies some specific requirements (see Definition 9) which facilitate an ordering and thus an easier search of solutions. Specifically, these requirements guarantee that any generated set of RCs can be ordered in such a way that when solved (within this ordering) each constraint becomes of the form

$$a_1 \dots a_n \sqsubseteq b_1 \dots b_m \gamma \tag{5}$$

where γ is the only variable.

Definition 8 {**Circular sequence of variables**} Let $\{RC_1, \dots, RC_m\}$ be a set of required constraints. A sequence of variables v_1, …, v_n is *circular* if there is an ordering RC_1, …, RC_m of the RCs such that for each $1 \leq i \leq n$ $v_i \in v_R(RC_i)$ and $v_{i+1} \in v_L(RC_i)$ and $v_n = v_j$ for some $j < n$.

For example, given the constraints $x_1v_2 \sqsubseteq y_1v_1$, $v_4x_2v_3 \sqsubseteq y_2v_2$, $x_3v_1 \sqsubseteq y_3v_3$, $x_4 \sqsubseteq v_4$, where x_i, for each $1 \leq i \leq 4$ and y_j, for each $1 \leq j \leq 3$, are sequences of non-variable atomic labels and v_h, for each $1 \leq h \leq 4$, are variables, the following two sequences of variables can be constructed v_1, v_2, v_3, v_1, and v_1, v_2, v_4 of which the first is a circular sequence.

Definition 9 {**Requirements on generated required constraints**} Let Γ be a derivation of a given labelled formula $A : z$, such that $\{RC_1, \dots, RC_m\}$ is the set of generated required constraints. Then, the following are *requirements* on this generated set of constraints.

1. There is exactly one variable on the RHS of each RC_i, where $1 \leq i \leq m$.
2. The constraint RC_j corresponding to the leftmost innermost sub proof of Γ satisfies $v_L(RC_j) = \emptyset$.
3. For each RC_i and each $v_i \in v_L(RC_i)$ there is exactly one $RC_{j(i)}$, with $j(i) \neq i$, such that $v_R(RC_{j(i)}) = v_i$. (The equality is justified by requirement 1.)
4. No *circular* sequence of variables can be formed from the given set of required constraints.

In particular, requirement (4) of the above definition implies that for each generated required constraint RC, $v_L(RC) \cap v_R(RC) = \emptyset$.

Theorem 2 *Any set of required constraints generated by the labelled natural deduction rules satisfies requirements (1)-(4) given in Definition 9.*

Proof outline: Each requirement is considered in turn and it is briefly explained why it is satisfied by the ND rules.

Requirement 1. The introduction of new variables on the RHS of a RC arise only using the $\otimes\mathcal{I}$ rule or the lemma rule. But since these rules always yield new constraints, their generated RCs will have only one variable on their respective RHSs – i.e. exactly the new variable introduced. The tick rule instead generates a RC between labels already introduced in the derivation. Hence, considering also that the label of the initial goal is a single new variable, the RHS of generated RCs have exactly one variable.

Requirement 2. The uppermost leftmost innermost box[2] always uses declarative units which are either temporary assumptions or derived from the initial assumptions by the $\rightarrow\mathcal{E}$ or $\otimes\mathcal{E}$ rules and therefore the generated labels are always ground. In both cases these labels occur on the LHS of the required constraints generated by either a $\otimes\mathcal{I}$ rule or a tick rule.

Requirement 3. Variables occurring in the LHS of a RC must have been already introduced in the structural derivation by applications of $\otimes\mathcal{I}$ or lemma rules. In both cases the associated RCs are new for each rule application. Therefore, there is always exactly one RC which contains in its RHS the variables used on the LHS of the constraints introduced in subsequent steps.

Requirement 4. Suppose that some circular sequence $\gamma_1, \gamma_2, \ldots, \gamma_i, \ldots, \gamma_i$ can be constructed from the generated RCs resulting from nesting of $\otimes\mathcal{I}$ and lemma rules. This implies, by Definition 8, that the variables introduced in some of these steps are not new, which is in contradiction with the definition of these rules.

□

The satisfiability of requirements (1)-(4) shown in the above theorem, allows the set of generated required constraints to be *ordered* in a list so that the left occurrences of any variable always appear in the list after the right occurrence. This is called an *ordered list.* This list is formed using the following process:

step 0 Find the set S_0 of required constraints such that for each $RC_i \in S_0$, $v_L(RC_i) = \emptyset$. (This set is non-empty by requirement (2) of Definition 9.) Let $R_0 = \bigcup_{RC_i \in S_0} v_R(RC_i)$.

step k**,** $k \geq 1$ Find the set S_k of required constraints such that for each $RC_i \in S_k$ $v_L(RC_i) \subseteq R_{k-1}$. Let $R_k = (\bigcup_{RC_i \in S_k} v_R(RC_i)) \cup R_{k-1}$.

The above process terminates when for some $k \geq 1$ the generated set S_k is empty and all the given required constraints have been considered, so giving the ordered list $S_0, S_1 \ldots, S_k$. The termination condition can be shown using the following reasoning by contradiction. Assume that for some k, $k \geq 1$, the associated set S_k is empty and that there are still some required constraints not considered in the process. Let S_r be the set of these required constraints ($S_r \neq \emptyset$). Then, by the definition of the above process this implies that for each $RC_i \in S_r$, $v_L(RC_i) \not\subseteq R_{k-1}$. Let RC_i be one of these constraints, with a variable $\gamma_1 \in \gamma_L(RC_i)$ and $\gamma_1 \notin R_{k-1}$. By requirement (3) there exists a constraint RC_j such that $\gamma_1 \in v_R(RC_j)$. Similarly, $v_L(RC_j)$ contains a variable γ_2 such that $\gamma_2 \notin R_{k-1}$ (else the constraint RC_j

[2] There may be several applications of the lemma rule in a proof, one beneath the other.

would belong to S_k contradicting the initial assumption). Continuing finding variables in this manner it is possible to form a sequence $\gamma_1, \gamma_2, \ldots, \gamma_n$. This sequence either stops by condition (i) of Definition 8, so yielding a $RC \in S_k$, or stops because a variable repeats itself. In the first case a contradiction arises with the initial assumption and in the second case a contradiction arises with the requirement (4) of Definition 9 and Theorem 2.

To summarise, any set of required constraints generated in a derivation can be ordered in a list where the required constraints on the leftmost part of the list will only have variables on their RHS, and the subsequent constraints will include these variables on their LHS and have new single variables on their respective RHS, which are themselves included in the LHS of the constraints coming next in the ordered list. The last constraint (i.e. the rightmost in the list) will have as its single RHS variable the label of the initial goal formula. Solving the constraints in the ordering direction "left-to-right" allows then instantiations of variables to be "propagated" throughout the list. The following sections describe how solutions of a given ordered list of required constraints can be found, and how the imposed constraints come into play.

4.2 Instantiating variables

The propagation of variables' instantiations between required constraints enables variables occurring on the LHS of a required constraint to be instantiated and the constraint itself to be reduced to a *simplified constraint* of the form given in (5), where only one variable occurs in the RHS. However, in the case of relevance and intuitionistic logics the simplified constraints have also to include auxiliary variables which are introduced in each required constraint either on the RHS or both on the LHS and RHS respectively in order to accommodate the additional properties of contraction and monotonicity. How to instantiate the variables of the resulting simplified constraints is shown below for each of the three logics.

4.2.1 Linear logic

Simplified constraints are of the form $a_1 \ldots a_n \sqsubseteq b_1 \ldots b_m \gamma$. To instantiate the variable γ it is sufficient to use the following *instantiation algorithm*: (i) check first that $b_1 \ldots b_m \subseteq a_1 \ldots a_n$ and (ii) if check (i) succeeds, define γ as

$$\gamma = a_1 \ldots a_n - b_1 \ldots b_m \tag{6}$$

4.2.2 Relevance and Intuitionistic logics

Slack variables: Simplified constraints include also auxiliary variables, called *slack variables*. In the case of relevance logic a slack variable is added to the RHS of a simplified constraint in order to cope with contraction. This is illustrated in the following example. Suppose that the RC to solve is $abbccc \sqsubseteq b\gamma$. Without applying contraction and adopting equation (6), γ has to be bound to $abccc$. But allowing contraction, there are also all the following possibilities: $\gamma = ac$, $\gamma = acc$, $\gamma = accc$, $\gamma = abc$, $\gamma = abcc$ and $\gamma = abccc$. To find all these solutions, a new variable δ_R is added to the RHS of the constraint. All the labels on the LHS that are not present on the right have to be *distributed* amongst γ and δ_R. In the case of intuitionistic logic, slack variables are added to the LHS and to the RHS of each required constraint to accommodate contraction and monotonicity.

In relevance and intuitionistic logics, the instantiation algorithms build upon the one given in linear logic.

Relevance logics: Simplified constraints are of the form $a_1 \dots a_n \sqsubseteq b_1 \dots b_m \gamma \delta_R$. The instantiation algorithm is analogous to that given in linear logic with the difference that the set-difference $a_1 \dots a_n - b_1 \dots b_m$ is *distributed* between γ and δ_R, with the *inclusion restriction* that if $x \in \delta_R$, then $x \in b_1 \dots b_m \gamma$. There is always a finite number of solutions.

Using this algorithm in the example given above, for each of the values of γ the following respective values of δ_R are found: bcc, bc, b, cc, c and 1. There are ten more candidate solutions, but none of them satisfies the inclusion restriction.

In general then, the instantiation of δ_R can only contain additional or *contracted* occurrences of a label – i.e. those that have already appeared elsewhere on the RHS. Notice that unique slack variables are added into each constraint and their values are not propagated as slack variables do not appear on the LHS of any RC.

Intuitionistic Logics: Simplified constraints are of the form

$$\delta_L a_1 \dots a_n \sqsubseteq b_1 \dots b_m \gamma \delta_R \tag{7}$$

To instantiate the variables γ, δ_R and δ_L it is sufficient to use the following *instantiation algorithm*:

(i) define δ_L as

$$\delta_L = b_1 \dots b_m - a_1 \dots a_n \tag{8}$$

(ii) define γ and δ_R as

$$\gamma \delta_R = \delta_L a_1 \dots a_n - b_1 \dots b_m \tag{9}$$

distributing the value of $\gamma \delta_R$ between γ and δ_R with the inclusion restriction that if $x \in \delta_R$, then $x \in b_1 \dots b_m \gamma$.

The instantiation of δ_L facilitates the construction of simplest solutions to equation (7). Consider the following example. Suppose that the constraint to solve is $ab \sqsubseteq \gamma bc$. One solution is $\gamma = a$. Another is $\gamma = ab$ and a more general solution is $\gamma = a\alpha$ (because of monotonicity), where α can be any combination of atomic labels. Now, the solution chosen for γ will in general be propagated into the LHS of other RCs. The addition of α will therefore itself be either propagated or contracted until all constraints have been solved. In the last constraint in the list the aim is for the original goal label to be instantiated to 1 if possible, otherwise to the simplest label possible. Restricting the solutions found in IL to be the simplest possible means, in the example, that α would be 1. This is achieved in general by the definition of δ_L given above. It is not difficult to check that the above assignments do yield solutions to constraints of the form given in (7) in IL, allowing for monotonicity and contraction. This is done by considering the numbers of each atomic label occurring in both sides of the constraints after substitutions are made for δ_L, δ_R and γ.

4.3 Using the Imposed Constraints

Imposed constraints are only generated by applications of the $\otimes\mathcal{E}$ rule to declarative units of the form $A \otimes B : x$. Therefore, they are always of the form $ab \sqsubseteq x$, where a and b are the solo parameter labels involved in the rule, and where x may or may not contain a variable. The use of an imposed constraint on a given required constraint is called *expansion* and it is defined as follows.

Definition 10 {**Expansion**} Let $ab \sqsubseteq x$ be an IC and let $(\delta_L)y \sqsubseteq z$ be a RC such that $y = aby_1$, where y_1 is an arbitrary sequence of labels. Then, the IC *expands* the RC into $(\delta_L)xy_1 \sqsubseteq z$. The RC is sometimes said to be *expanded* into $(\delta_L)xy_1 \sqsubseteq z$, and the latter is sometimes called the *expanded constraint.*

If an expanded constraint $xy_1 \sqsubseteq z$ is satisfied then the original required constraint of the form $aby_1 \sqsubseteq z$ is also satisfied since $aby_1 \sqsubseteq xy_1 \sqsubseteq z$.

To enforce that the LHS (apart from δ_L if present) of a generated expanded constraint remains ground, only ICs with ground RHSs should be used in an expansion step. As far as this is concerned, ICs of the form $ab \sqsubseteq x$, where x contains a variable, do not need to be used in an expansion step until x is ground. This is justified by the following observation. Any variable α in the RHS of an IC must have been derived from some application of the lemma rule. A system similar to the one described here could be defined by introducing in the labelling language an additional operator called the *residual* operator and denoted with /, and using it in the $\otimes\mathcal{E}$ rule. The labelling algebra is extended with the characteristic property of / given in (10)

$$y \circ z \sqsubseteq x \quad \text{iff} \quad z \sqsubseteq x/y \tag{10}$$

In such a system, instead of deriving $TA : a$ and $TB : b$, where $a \circ b \sqsubseteq x$, from $TA \otimes B : x$, $TA : a$ and $TB : x/a$ would be derived, without the generation of the IC. In this case there would be no corresponding IC to use in the derivation of the sub-proof of a lemma rule application, so that the RC involving the variable α on its RHS would be solved without the IC. In the system described in this paper, the same would be true, showing that an IC needs only to be used to expand the LHS of a RC after it has been introduced, and hence after its RHS has become ground. However the use of the / would have made the whole solving process more complex.

It can be seen that the use of ICs increases the possibilities for finding solutions. For example, in LL suppose that, in addition to the RC $abbccc \sqsubseteq \gamma b$, there is the IC $bc \sqsubseteq d$. Then, there are two solutions: either after 0 expansion steps, $\gamma = abccc$, or after 1 expansion step, $\gamma = adcc$. Note also that a RC such as $bca \sqsubseteq \gamma d$ can only be solved by using the IC first, to derive $da \sqsubseteq \gamma d$.

In RL, contraction might be needed when applying an IC. For example, it is needed in order to solve the following system of constraints in RL: $ab \sqsubseteq c$ as an imposed constraint and $abb \sqsubseteq c$ as a required constraint. This is implemented in the same spirit as the instantiation step, by including a slack variable δ_L on the left of the IC. Thus, in RL an imposed constraint $ab \sqsubseteq x$ becomes $ab\delta_L \sqsubseteq x$ and expansion matches $ab\delta_L$ instead of ab, where the restriction $\delta_L \subseteq \{a, b\}$ is imposed.

In IL, $ab \sqsubseteq x$ implies $a \sqsubseteq x$, for $a \sqsubseteq ab \sqsubseteq x$, the first step following from monotonicity. Similarly, $b \sqsubseteq x$ is also implied. Hence, in IL an IC of the form $ab \sqsubseteq x$ can be used in an expansion step in three different ways: either to replace an occurrence of ab by x in the RHS of a RC, or to replace an occurrence of a or an occurrence of b by x in the RHS of a RC. This reflects the fact that the $\otimes$ operator in IL behaves exactly as the usual classical conjunction.

4.4 The algorithm

Putting the variable instantiation procedure of Section 4.2 together with expansion, yields the following algorithm for solving a label constraint problem, called *the solving process*:

Solving process:
Suppose that $\langle IC_1, \ldots, IC_n, RC_1, \ldots, RC_m \rangle$, $m \geq 1$, $n \geq 0$ is the constraint problem to be solved, in which the RCs are ordered as described in Section 4.1. In order to solve the RCs, using the ICs, the following two types of steps are made:

(i: instantiation) in which the variables in a required constraint are instantiated as described in Section 4.2;

(ii: expansion) in which an IC is applied to the LHS of a RC (employing the contraction and monotonic properties as appropriate).

It may be possible to make an arbitrarily large number of expansion steps to a particular required constraint (see for example the problem illustrated in Figure 12). Therefore, in order for the solving process to terminate, some kind of limit must be placed on the number of expansion steps made. To better approximate all solutions an incremental limit should be used. Further investigations of a possible upper limit are currently being undertaken.

5 Correctness of the Natural Deduction Rules

In this section the set of labelled natural deduction rules described in Table 2 is shown to be sound with respect to the LKE tableaux system described in [DG94]. This is proved by firstly extending the LKE system with an additional rule, called (Tch), secondly showing that this extension is equivalent to the original LKE system and then proving that for each natural deduction derivation there exists a corresponding extended LKE refutation. Before going into the details of the proof a brief description of the LKE rules is given.

5.1 The LKE system

The LKE system described in [DG94] is a uniform labelled semantic tableaux system for substructural logics which generalises the classical logic KE-tableau system [DM94] using the LDS approach. Within this system, the refutation rules are common to any substructural logics – they perform the role of operational rules. The standard structural rules of substructural logics are expressed in terms of conditions on the labels, so different logics are captured by just considering different labelling algebras. Note that in [DG94] a labelling algebra is defined in terms of a complete lattice. However, the correspondence of theorems in LKE with respect to a class of labelling algebras associated with a logic Δ, with theorems in the logic Δ shows that, in the multiplicative fragment, only labels constructed from atomic labels appearing in an LKE tree and the $\circ$ operator are necessary. The rule for closing branches takes into account these conditions allowing then some formulae instead of others to be proved. Within a LKE system, a formula A is proved to be a theorem of a given substructural logic if it is possible to show that there exists a refutation of the assumption "A being false at the identity element 1" for the class of labelling algebras associated with the given logic. Such a refutation is a LKE tree starting with the labelled signed formula $FA : 1$, and having all the branches closed by applications of the (Cl) rule. In this system there is no clear distinction between semantics and syntax. Semantic notions of a labelling algebra and its consequences, such as the existence of characteristic labels, are integrated into the proof system.

The set of refutation rules, given in [DG94] and restricted to the fragment $\{\rightarrow, \otimes\}$ of substructural logics, is listed in Table 3. The rules for the operators $\rightarrow$ and $\otimes$ can be proved

$$(T\rightarrow)\ \frac{\begin{array}{l} TA \rightarrow B : x \\ TA : y \end{array}}{TB : x \circ y} \qquad (F\rightarrow)\ \frac{FA \rightarrow B : x}{\begin{array}{l} TA : a \\ FB : x \circ a \end{array}} \ \text{where } a \text{ is the } A\text{-}\textit{characteristic} \text{ atomic label}$$

$$(T\otimes)\ \frac{TA \otimes B : x}{\begin{array}{l} TA : a \\ TB : x/a \end{array}} \ \text{where } a \text{ is the } A\text{-}\textit{characteristic} \text{ atomic label} \qquad (F\otimes)\ \frac{\begin{array}{l} FA \otimes B : x \\ TA : y \end{array}}{FB : x/y}$$

$$(\text{PB})\ \frac{}{FA : x \mid TA : x} \qquad (\text{Cl})\ \frac{\begin{array}{l} TA : x \\ FA : y \end{array}}{\times} \ \text{provided } x \sqsubseteq y$$

Table 3: The LKE rules for the substructural fragment $\{\rightarrow, \otimes\}$.

[DG94, BDR96] to be respectively equivalent to the clauses (3) and (4) of Definition 2 in Section 2. In the case of $(F \rightarrow)$, the use of the A-characteristic atomic label a is justified by the following reasoning. If x does not verify $A \rightarrow B$ then by clause (3) of Definition 2 there exists some label which verifies A and which composed with x does not verify B. This implies by the same algebraic property [3] given in clause (2) of Definition 1, that there exists a least label, the A-characteristic, denoted with a, which verifies A. The same argument holds for the $(T\otimes)$ rule, but with respect to the semantic clause (4) of Definition 2. Notice that in the rules for the multiplicative conjunction $\otimes$, the operator "/" defined in (10) is used. The (PB) rule expresses the labelled version of the semantic *Principle of Bivalence* which states that any formula A can be either true or false at each element of the labelling algebra. Application of the (PB) rule can be restricted to sub-formulae of the formulae appearing in the tree. The

[3] This is a consequence of the definition of labelling algebra used in [DG94].

version used here employs free variables, as described in [DG94]. The (Cl) rule instead states when a branch can be closed, and it is justified by clause (1) of Definition 2 given in Section 2. The application of this rule depends on the side condition $x \sqsubseteq y$, which can be proved to hold using the properties of $\circ$ of the underlying class of labelling algebras. However, the work in [DG94] does not cover the issue of finding algorithms for solving these kind of conditions. The first and only example has been given in [BF95] but only for the case of Linear Logic.

Extending LKE The LKE system is here extended by adding an extra rule to those given in Table 3, called (Tch):

$$(Tch)\ \frac{TA : x}{TA : a} \quad \text{where } a \text{ is the } A\text{-} \textit{characteristic} \text{ atomic label} \tag{11}$$

This rule reflects the algebraic property given in clause (2) of Definition 1 and will provide the link between the use of the / operator in the LKE system and the use of the imposed constraints in the natural deduction system. Specifically, introduction of a label using / in the LKE system by the (T⊗) rule, corresponds to the introduction of an IC in the natural deduction system. Solo parameters of L_L used in a natural deduction derivation are mapped into the LKE characteristic atomic label.

The extended LKE system is equivalent to the original LKE system. This is illustrated in Figure 4 and explained below.

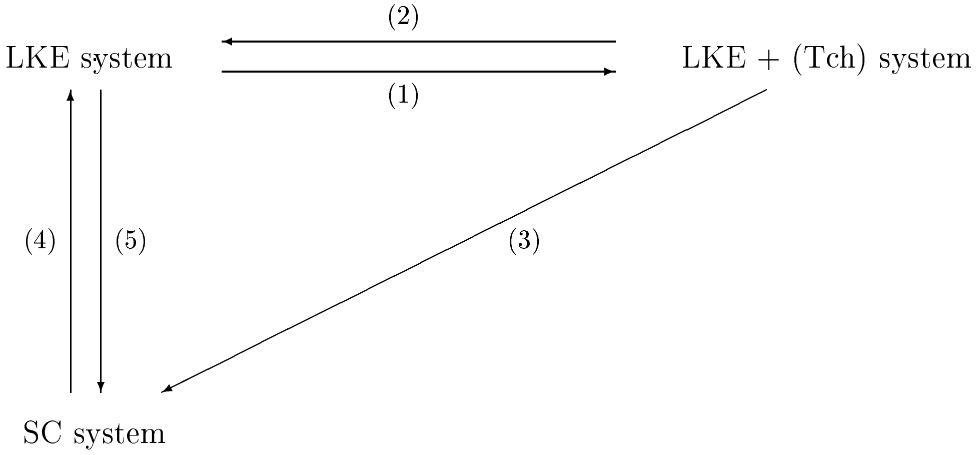


Figure 4: Equivalence between the Extended LKE and the original LKE tableaux system.

Arrows 4 and 5 in Figure 4 describe the completeness and soundness property of the LKE system with respect to the sequent calculus, proved in [DG94] by Propositions 4 and 5 respectively. Arrow 1 shows that for any refutation in the original LKE system there is a refutation in the extended LKE system. This is trivially true. Hence to show the equivalence between the two LKE systems (original and extended) it is just sufficient to show that the extended LKE system is sound with respect to the sequent calculus (arrow 3). This is proved by extending Proposition 5 of [DG94] to the case of the (Tch) rule. The proof of Proposition 5 in [DG94] is based on a *canonical interpretation*, which interprets labels as sets of formulae closed under the sequent calculus derivability relation and declarative units of the form $TA : x$ as $A \in x$, and on a canonical valuation v defined as $v(A, x) = T$ iff $A \in x$. Under this canonical interpretation $TA : x$ holds iff $v(A, x) = T$. Extending Proposition 5 to cover the (Tch) rule requires to show that if $TA : x$ holds then $TA : a$ holds, where a is the A-characteristic element in the labelling algebra. Using the above canonical interpretation $TA : x$ holds implies $v(A, x) = T$, which, by clause (2) of Definition 1, implies that $v(A, a) = T$ and hence $TA : a$ holds. The reader is referred to [DG94] for further details.

Completeness with respect to LKE In [BDR96] it has been proved that this system of natural deduction rules is complete with respect to the LKE system. That is, there exists an algorithm which turns a LKE refutation of a theorem $A : 1$, formed in a particular way, into a labelled natural deduction derivation of $A : 1$, showing also that tableau refutation rules can be read as backward reasoning in this labelled natural deduction system. For more details the reader is referred to [BDR96].

5.2 Soundness with respect to LKE

The soundness of the ND system with respect to LKE is shown in this section, but with respect to the extended LKE system. The equivalence between the LKE system and the extended LKE system implies the soundness of the ND rules with respect to the LKE system. In the rest of this section "LKE system" will refer to this extended set of LKE rules.

In the following proof, notions of "lengths of the ND rules" and "length of a derivation" are used. These are defined as follows. The ND rules which do not have sub-derivations in their antecedents, with the exception of the $\otimes\mathcal{E}$ rule, have length equal to 1 (i.e. these are the Tick rule and the $\rightarrow\mathcal{E}$ rule), the $\otimes\mathcal{E}$ rule has length equal to 2, and the rest of the ND rules have length equal to the (sum of the) length(s) of the smallest subderivation(s) in their antecedent incremented with 1. For example, the $\rightarrow\mathcal{I}$ rule has length equal to $1 + l_1$ where l_1 is the length of the smallest derivation in its antecedent, whereas the length of the $\otimes\mathcal{I}$ rule is equal to $1 + l_1 + l_2$ where l_1 and l_2 are respectively the lengths of the smallest left-hand-side and right-hand-side derivations in the rule's antecedent. The length of a given arbitrary derivation is thus equal to the sum of the lengths of the inference rules used in that derivation. Notice that, as in the standard natural deduction, a structural derivation with length equal to 0 is a derivation with no inference rule application, with an empty set of constraints and with the goal being a declarative unit already belonging to the given set of initial assumptions.

The proof of Theorem 3 uses a mapping between ND derivations and LKE trees defined as follows. Each label f of assumptions of the form $F : f$ which occurs in a ND derivation, is mapped to a corresponding label f in a signed formula $TF : f$ where f is now the F-characteristic label. For any other non atomic label the mapping is extended in an obvious way. The restriction imposed on the solo parameters in the ND rules guarantees this mapping to be a one-to-one function.

Theorem 3 {**Soundness**} *Let Δ be a given substructural logic, let T be a set of initial assumptions of the form $\{A_1 : a_1, \ldots, A_n : a_n\}$ and let $B : z$ be a declarative unit. If $T \vdash_\Delta B : z$ then there exists a closed LKE tableau refutation for $TA_1 : a_1$, ..., $TA_n : a_n$, $FB : z$.*

Proof: Let $\langle\{\Gamma_1, \ldots, \Gamma_k\}, \{IC_1, \ldots, IC_n, RC_1, \ldots, RC_m\}\rangle$, with $k \geq 1$ and $n, m \geq 0$, be the smallest derivation of $B : z$ in the logic Δ with length l. The proof is by induction on l.

Base Case

The base case is when $l = 0$, i.e. there is no inference rule application. Therefore for some $1 \leq i \leq n$, $B : z = A_i : a_i$ (i.e. $B = A_i$ and $z = a_i$). Hence, there is also a closed LKE tableau refutation for the set $TA_1 : a_1$, ..., $TA_n : a_n$, $FB : z$, given by one application of the (Cl) rule between $A_i : a_i$ and $B : z$.

Inductive Step

Suppose that $l > 0$ and that the theorem holds for any smallest derivation of length less than l. The proof is then by cases for any rule application on the last step which generates Γ_k. (For simplicity, in each of these cases the initial theory T is subsumed in both the natural deduction derivations and the corresponding LKE trees.)

Case 1: Tick rule.

In this case $\Gamma_k = \langle B : z, \checkmark\rangle$, a declarative unit of the form $B : x$, for some label x, is part of the derivation $\{\Gamma_1, \ldots, \Gamma_{k-1}\}$, $x \sqsubseteq z \in \{RC_1, \ldots, RC_m\}$ and the associated set of label constraints $\{IC_1, \ldots, IC_n, RC_1, \ldots, RC_m\}$ is satisfied in Δ. A LKE refutation of $FB : z$ can be constructed starting with $FB : z$ and the set of initial assumptions T. A (PB) rule is applied on $B : x$. The left branch closes by the inductive hypothesis using the part of the derivation denoted by Π_1, which is the part of the ND derivation which has inferred $B : x$. A labelled signed formula of the form $TB : x$ occurs in the right branch, which then closes with the application of a (Cl) rule. The condition of the (Cl) rule holds by hypothesis.

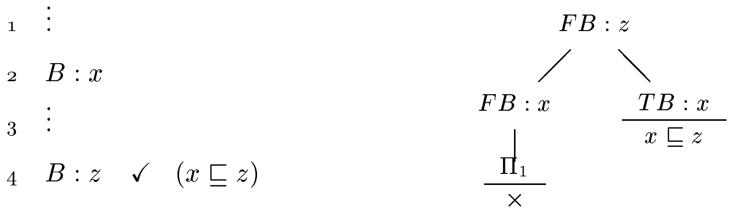


Figure 5: Correspondence between labelled ND and LKE for the Tick rule.

Case 2: $\rightarrow\mathcal{E}$.

In this case $\Gamma_k = \langle B : x \circ y, \rightarrow\mathcal{E}\rangle$ (i.e. $z = x \circ y$), declarative units of the form $D \rightarrow B : x$ and $D : y$, for some labels x and y are part of the derivation $\{\Gamma_1, \ldots, \Gamma_{k-1}\}$, and the set of constraints $\{IC_1, \ldots, IC_n, RC_1, \ldots, RC_m\}$ is satisfied in Δ. A LKE refutation of $FB : x \circ y$ can be constructed which starts with $FB : z$ and the set of initial assumptions T. A (PB) rule is applied on $D \to B : x$. The left branch closes by the inductive hypothesis using part of the derivation denoted with Π_1, which is the part of the ND derivation which has inferred $D \to B : x$ and whose length is less than L. On the right branch another (PB) rule is applied on $D : y$. Its left branch closes by inductive hypothesis using part of the derivation denoted with Π_2, whereas its right branch closes with the application of $(T \to)$ and (Cl) rules as shown in Figure 6.

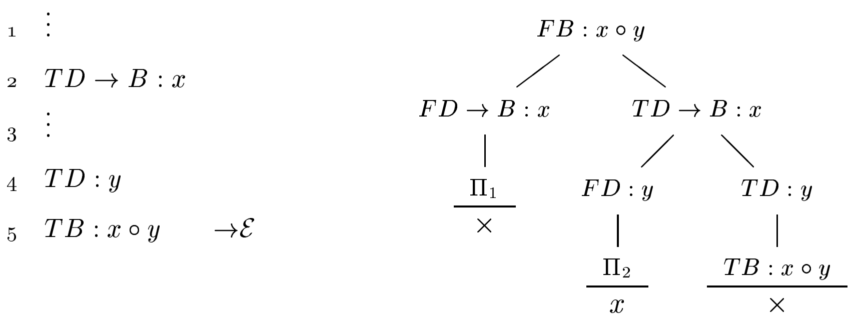


Figure 6: Correspondence between labelled ND and LKE for the $\to\mathcal{E}$ rule.

Case 3: $\otimes\mathcal{E}$.

In this case a declarative unit of the form $D \otimes B : x$ for some label x, is part of the derivation $\{\Gamma_1, \ldots, \Gamma_{k-1}\}$, Γ_k is either equal to $D : d$ or $B : b$ where d and b are solo parameters; the constraint $d \circ b \sqsubseteq x \in \{IC_1, \ldots, IC_n\}$, and the whole set of constraints $\{IC_1, \ldots, IC_n, RC_1, \ldots, RC_m\}$ is satisfied in Δ. Consider the first case. A LKE refutation of $FD : d$ can be constructed by starting with $FD : d$ and applying a (PB) rule on $D \otimes B : x$. The left branch closes by inductive hypothesis using part of the derivation denoted with Π_1, which has inferred $D \otimes B : x$ The right branch closes after the application of a $(T\otimes)$ rule. This is shown in Figure 7. The second case, when $\Gamma_k = B : b$, can be proved using an analogous argument, except that this time $TB : b$ is derived in the right branch from $TB : x/d$ by the (Tch) rule. This step may appear to use the hereditary property illegally, but remember it is an application of the (Tch) rule.

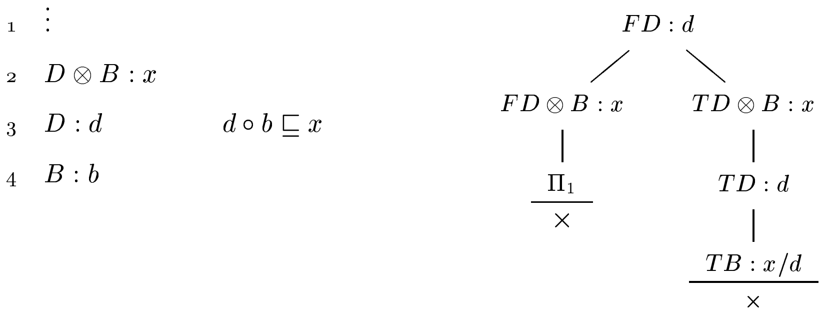


Figure 7: Correspondence between labelled ND and LKE for the $\otimes\mathcal{E}$ rule.

Case 4: $\to\mathcal{I}$ In this case the tuple Γ_k is of the form $\langle B \to C : z, \to\mathcal{I}\rangle$. The last step is therefore an $\rightarrow\mathcal{I}$ rule. A LKE refutation of $FB \rightarrow C : z$ can be constructed by starting with $FB \rightarrow C : z$, adding the assumption $TB : b$, applying the $(F \rightarrow)$ and by applying for each preceding ND rule the corresponding LKE rule. (This last step is denoted with Π_1 in Figure 8). The LKE tree closes by inductive hypothesis on Π_1.

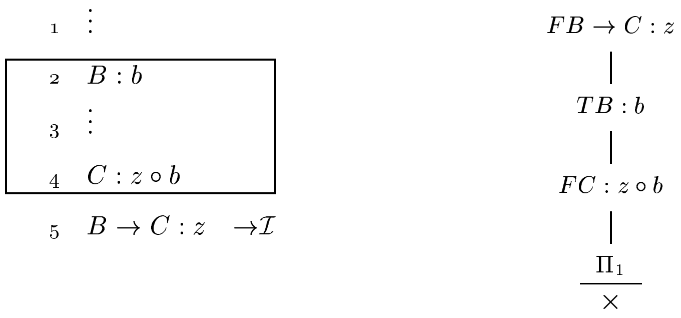


Figure 8: Correspondence between labelled ND and LKE for the $\rightarrow\mathcal{I}$ rule.

Case 5: $\otimes\mathcal{I}$. In this case the tuple Γ_k is of the form $\langle B \otimes C : z, \otimes\mathcal{I}\rangle$. A LKE refutation of $FB \otimes C : z$ can be constructed in the way described below. Start with $FB \otimes C : z$ and apply a (PB) rule on $B : \gamma_1$. The left branch closes by inductive hypothesis on the part of the ND derivation which proves $B : \gamma_1$ (denoted with Π_1 in Figure 9). In the right branch a $(F\otimes)$ rule is applied deriving $FC : z/\gamma_1$. By inductive hypothesis there exists an LKE refutation Π_2 of $FC : /\gamma_2$. Since $\gamma_1 \circ \gamma_2 \sqsubseteq z$ and hence $\gamma_2 \sqsubseteq z/\gamma_1$ there is also a refutation of $FC : z/\gamma_1$.

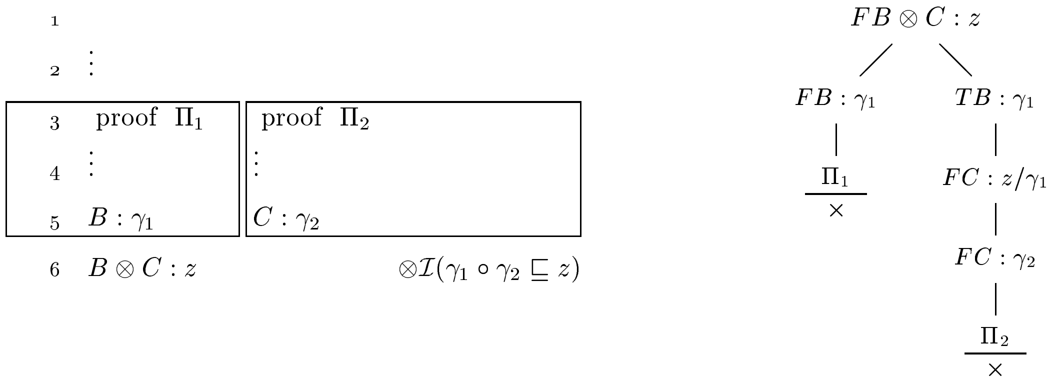


Figure 9: Correspondence between labelled ND and LKE for the $\otimes\mathcal{I}$ rule.

Case 6: Lemma rule.

In this case the tuple Γ_k is of the form $\langle B : z,$ Lemma $\rangle$. A LKE refutation of $FB : z$ can be constructed by starting with $FB : z$ and making an application of (PB) with $B : z$. The left branch closes by the inductive hypothesis using part of the derivation denoted by Π_1, which is the part of the ND derivation which has inferred $B : z$. On the right branch the tree closes immediately.

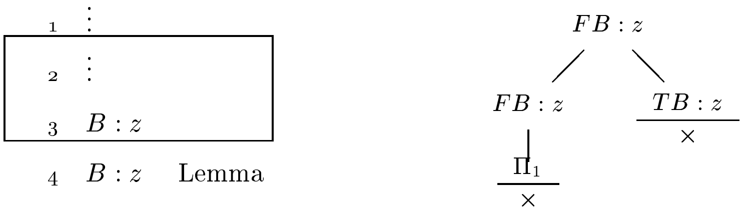
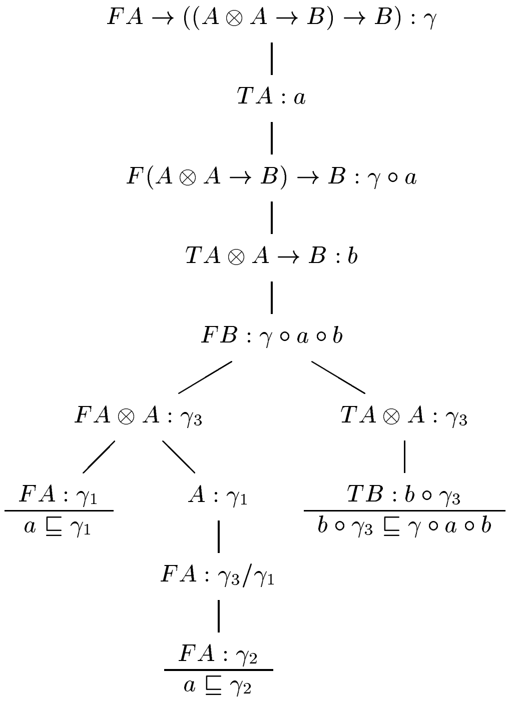


Figure 10: Correspondence between labelled ND and LKE for the Lemma rule.

□

As an example of the correspondence between natural deduction and the extended LKE system Figure 11 shows the tableau corresponding to the natural deduction derivation given in Figure 1, but with γ replacing each occurrence of 1 in the labels. In this LKE refutation the rightmost branch closes using the required constraint $b \circ \gamma_3 \sqsubseteq \gamma \circ a \circ b$. The left branch under the node $FA \otimes A : \gamma_3$ closes using the required constraint $a \sqsubseteq \gamma_1$, whereas the other branch closes using the required constraint $a \sqsubseteq \gamma_2$.



$FA \to ((A \otimes A \to B) \to B) : \gamma$
$TA : a$
$F(A \otimes A \to B) \to B : \gamma \circ a$
$TA \otimes A \to B : b$
$FB : \gamma \circ a \circ b$
$FA \otimes A : \gamma_3$
$TA \otimes A : \gamma_3$
$FA : \gamma_1$
$a \sqsubseteq \gamma_1$
$A : \gamma_1$
$TB : b \circ \gamma_3$
$b \circ \gamma_3 \sqsubseteq \gamma \circ a \circ b$
$FA : \gamma_3/\gamma_1$
$FA : \gamma_2$
$a \sqsubseteq \gamma_2$

Figure 11: LKE tree corresponding to the ND derivation of Figure 1.

6 Conclusion

This paper has shown how a uniform proof method for substructural logics based on natural deduction can be defined using the LDS approach. This system is sound and complete with respect to the LKE system. These properties, together with the correspondence of the LKE system with respect to the sequent calculus, proved in [DG94], implies that the natural deduction is also sound and complete with respect to the sequent calculus for the class of substructural logics described in [DG94].

However, the two proof theoretic approaches (ND and LKE system) present significant differences. This is briefly discussed in the following section where an example illustrating such differences is also given.

6.1 Comparison between the ND and LKE systems

One of the features of the natural deduction approach described in this paper is that for a given derivation many different solutions for the variable label of the initial goal can be found which satisfy the associated set of label constraints. This is due to the fact that in the solving process no limit is fixed "a priori" to the number of expansion steps which can be applied to a given set of required constraints. Often, additional applications of the expansion step lead to additional solutions, for the same associated structural derivation. In the LKE system, no use is made of the imposed constraints. The expansion step of the ND's solving process, which uses a generated IC, corresponds instead to additional applications of LKE refutation rules. Consequently, a single ND structural derivation can correspond to more than one (and possibly to an infinite number of) closed standard LKE trees[4]. This is shown in the following example.

Consider the ND derivation, given in Figure 12, of the declarative unit $E : \gamma$ from the set of initial assumptions $\{A : a, \quad A \rightarrow D : d, \quad D \rightarrow E : e, \quad (A \rightarrow D) \rightarrow (D \rightarrow E) \rightarrow (A \rightarrow D) \otimes (D \rightarrow E) : c\}$. The generated set of ICs is given by $\{de \sqsubseteq cde\}$ and the set of required constraints is $\{eda \sqsubseteq \gamma\}$. (In this example, the operator $\circ$ is omitted for simplicity.) In the case of LL, there are many solutions for the variable γ which satisfy the label constraints, namely $\gamma = eda$, $\gamma = ecda$, $\gamma = eccda$ and so on. Each of these solutions is obtained by making use of the imposed constraint $de \sqsubseteq cde$ none, one or more times. (In RL and IL all solutions which have at least one occurrence of c are equivalent because of the contraction property.) Notice that if in this example the variable γ had been the particular label $adeccf$, for some arbitrary initial assumption $F : f$, then no solution could have been found; yet the imposed constraint $de \sqsubseteq cde$ would have been applied infinitely many times to the label eda, but never yielding $adeccf$. This shows once more the semi-decidability of the solving process algorithm.

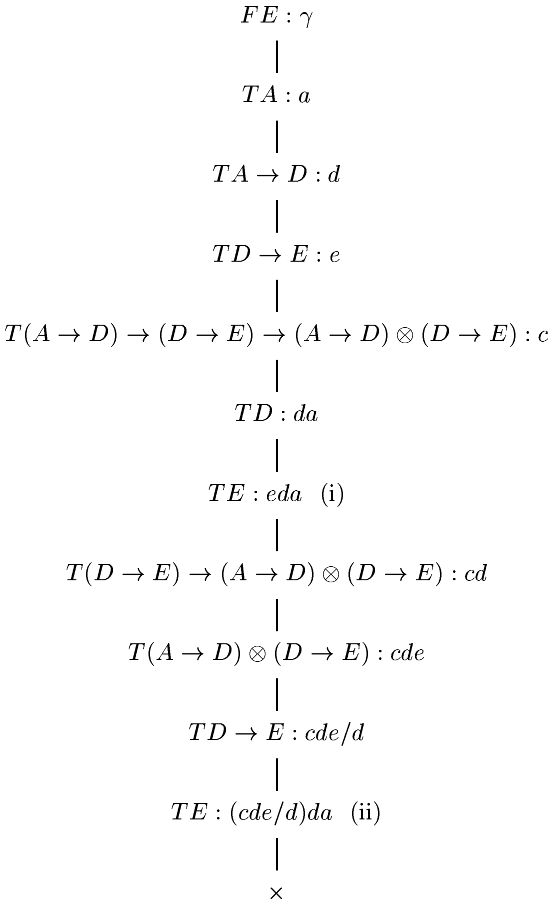
When a comparison is made with the standard LKE system, it is seen that a single natural deduction structural derivation, such as the one on the left in Figure 12, can correspond to more than one (and possibly to an infinite number of) closed "standard" LKE-trees. Examples of two LKE refutations are given, within one tree in the right-hand diagram of Figure 12. These are obtained by terminating the tree at (i) and (ii) respectively. In this example,

[4] Standard LKE-trees are LKE trees generated using only the set of rules described in Table 3.

1	$A : a$	
2	$A \to D : d$	
3	$D \to E : e$	
4	$(A \to D) \to (D \to E) \to (A \to D) \otimes (D \to E) : c$	
5	$(D \to E) \to (A \to D) \otimes (D \to E) : cd$	$(2, 4)$
6	$(A \to D) \otimes (D \to E) : cde$	$(3, 5)$ $(*)$
7	$D : da$	$(1, 2)$
8	$E : eda$	$(1, 2, 3)$
9	$E : \gamma$	✓ $eda \sqsubseteq \gamma$

$(*)$ IC= $de \sqsubseteq cde$ from (6), (2) and (3)

$A \to D : d$ and $D \to E : e$
are already present by step (6) and so are not added again.



$FE : \gamma$
$TA : a$
$TA \to D : d$
$TD \to E : e$
$T(A \to D) \to (D \to E) \to (A \to D) \otimes (D \to E) : c$
$TD : da$
$TE : eda$ (i)
$T(D \to E) \to (A \to D) \otimes (D \to E) : cd$
$T(A \to D) \otimes (D \to E) : cde$
$TD \to E : cde/d$
$TE : (cde/d)da$ (ii)
$\times$

Figure 12: Example of structural derivation and standard LKE-tree(s)

supposing that the underlying logic is LL, each solution of the variable label γ corresponds to a different standard LKE-tree, depending on the number of times the signed labelled formula $T(D \to E) \to (A \to D) \otimes (D \to E) : cd$ (shortened to TX) is used. The first solution, $\gamma = eda$, corresponds to a tree in which such a signed labelled formulae TX is used no times (i.e. in this case closure is made at the step (i)); on the other hand the second solution, $\gamma = ecda$, corresponds to a tree in which the signed labelled formula TX is used once and the derived signed labelled formula $D \to E : cde/d$ is used to derive $E : (cde/d)da$ (in this case closure is made at step (ii)). Further solutions correspond to bigger trees[5].

This correspondence between the ND system and the LKE system implies that if for a given structural derivation the solving process does not terminate (e.g. there is no solution to a given set of label constraints) the set of LKE trees reflecting the natural deduction structural derivation would also not terminate.

Additional observations about the two systems are: (i) use of free variables is made in both

[5] For example, the third solution corresponds to a tree in which the signed labelled formula TX is used twice, and in which $T(A \to D) \otimes (D \to E) : cd(cde/d)$ and then $D \to E : (cd(cde/d))/d$ are derived and then finally $TE : ((cd(cde/d))/d)da$ is derived. Using $e \sqsubseteq cde/d$, associativity of $\circ$ and the simplification rule for the / operator, namely $xz(y/z) \sqsubseteq xy$, it can be shown that $(cde/d)da \sqsubseteq ecda$ and $((cd(cde/d))/d)da \sqsubseteq cd(cde/d)a \sqsubseteq eccda$.

systems (i.e. in the ND Lemma rule and in the LKE (PB) rule) and (ii) limitations, as to the number of times rules are used in a proof, are still to be thoroughly investigated in both systems. As far as (i) is concerned, in [DG94] it has been observed that the use of free variables in the (PB) rule can vastly improve the practicality of the method. That is, instead of "guessing" the value of the label x to use in a (PB) rule application, a free variable γ can be used; γ is treated within the proof as a ground label, and it is only at the closure step that a suitable value for it is given using the closure inequation. The soundness and completeness of LKE, with respect to the sequent calculus, shows that only values for γ involving $\circ$ and $/$ are necessary. This corresponds to the use of a free variable in the ND Lemma rule. However in the ND system this approach is taken even further, allowing free variables to be used also in the $\otimes\mathcal{I}$ rule. Simple label inequations involving only the $\circ$ operator generated by the closure rule can be solved in the LKE system using algorithms based on the AC-unification technique[Sti81]. (See [BF95] for further details.) For more complex inequations involving the / operator no algorithm has, to the authors' knowledge, yet been reported. In the ND approach, the solving process is much simpler. ND proofs are more structured. This structural feature facilitates the definition of an ordering procedure on the generated label constraints which simplifies the instantiation process and therefore the search for solutions.

However, two important practical difficulties still remain when incorporating free variables into the LKE method. The first one is how to limit the applications of the rules and still retain completeness. (For example, (T$\rightarrow$) may be applied indefinitely to $A \rightarrow A : y$ given $A : x$.) In [DG94], it is stated, but not proven, that the free variable version of the (PB) rule only need to be used at most once for each occurrence of a T$\rightarrow$ or F$\otimes$ signed labelled formula. This restriction together with an examination of the labels of the F-formulas in an LKE-tree, which may allow restrictions on the labels of T-formulas derived from applications of (T$\rightarrow$), can be used to impose finiteness on the LKE-trees. The second difficulty is concerned with algorithms to solve general inequalities involving the / operator. In the LKE approach, the equivalence (10) given in Section 4.3 may be used to derive some additional general properties, such as $x \circ z \circ (y/x) \sqsubseteq z \circ y$ and $(x/y)/z = x/(yz)$. These properties could be used to solve constraints between terms involving the / operator. Nevertheless, label inequalities still remain difficult to solve. In the ND approach, the difficulty of handling terms with the / operators is avoided by the introduction of imposed constraints. In fact, applications of an imposed constraint in an expansion step of the solving process could be seen as using the general rule $x \circ z \circ (y/x) \sqsubseteq z \circ y$ on an LKE constraint involving the / operator. The search for a limit on the number of rules applications is instead a difficulty for the ND approach as well.

6.2 Final remarks

The method described here provides a way of carrying out natural deduction proofs for the three most well known Substructural Logics, namely Linear Logic, Relevance Logic and Intuitionistic Logic. To cover the cases of other substructural logics which already exist in the literature or which may be proposed in the future, it is sufficient to appropriately adapt the solving process algorithm, leaving unchanged the set of labelled ND rules given in Table 2. For example, in the case of Lambek calculus the simple instantiation algorithm used for Linear logic can be strengthened to deal with lists rather than multisets in order to avoid the commutativity property. The case of Mingle's implication can be obtained by restricting the monotonicity property to the expansion property (i.e. $x \sqsubseteq x \circ x$). This latter case can be dealt with by appropriately adapting the instantiation step of the solving process algorithm for intuitionistic logic.

In this paper the description is limited to the syntactical fragment of substructural logics containing only material implication and the multiplicative conjunction. Multiplicative negation can be easily incorporated into the system for classical substructural logics. It would be sufficient to translate negated formulae of the form $\neg A$ into $A \rightarrow \bot$, where $\bot$ is a constant proposition representing falsity, and introduce the following rule (corresponding to double negation elimination):

$$\frac{T(A \rightarrow \bot) \rightarrow \bot : x}{TA : x} \tag{12}$$

In the presence of negation, $A \otimes B : x$ can be equivalently translated into $\neg(A \rightarrow \neg B) : x$. This removes the necessity for the expansion step in the algorithm, leading to guaranteed termination of the solving process for a particular structural derivation and therefore a one to one correspondence of structural derivations with LKE trees (not using $\otimes$ either). For any of the above extensions as well as for the system described in this paper, the decidability property of the system needs still to be proven. This is also the case of the LKE system as well as of any other proof theoretic approach to substructural logics. It could be foreseen however that to obtain such a result additional controls are needed to restrict (i) the number of lemma rule applications, (ii) the number of $\rightarrow\mathcal{E}$ application and (iii) the number of expansion steps in the solving process. This is currently under investigation and it is believed that the structural feature of the natural deduction proofs could help in finding such proof theoretic restrictions.

Finally, the issue of uniformity poses one interesting question: What is the price paid for such generality? It is known that the multiplicative fragment of LL (i.e. $\{\otimes, \rightarrow, \neg\}$) is NP-complete. The algorithm described here is EXP-time for the $\{\rightarrow, \otimes\}$ fragment of LL and RL. This appears to be a reasonable complexity, given the theoretical lower bounds. It could be argued that in a theorem prover for LL only, special heuristics could be developed which would improve its efficiency. But this could be done only making the theorem prover specific to one particular logic. In the ND system described in this paper, the modularity of the derivation process given by the use of the labels, would facilitate such heuristics to be embedded into the constraint solving mechanism, leaving the set of ND rules general and applicable to any substructural logic. There is always space then for efficiency gains.

Acknowledgements

The authors would like to thank Marcello D'Agostino and Hans-Jorgen Ohlbach for their helpful comments and also several anonymous referees for their suggestions for improvements.

References

[BDR96] Krysia Broda, Marcello D'Agostino, and Alessandra Russo. Transformation methods in LDS. In *Logic, Language and Reasoning. An Essay in Honor of Dav Gabbay.* Kluwer

Academic Publishers, To appear 1996.

[BEKV94] K. Broda, S. Eisenbach, H. Khoshnevisan, and S. Vickers. *Reasoned Programming*. Prentice Hall, 1994.

[BF95] Krysia Broda and Marcelo Finger. KE-tableaux for a fragment of linear logic. Technical report, 4th Workshop on Theorem Proving with Analytic Tableaux and Related Methods, Ed. Peter Baumgartner, University of Koblenz, 1995.

[DG94] Marcello D'Agostino and Dov M. Gabbay. A generalization of analytic deduction via labelled deductive systems.Part I: Basic substructural logics. *Journal of Automated Reasoning*, 13:243–281, 1994.

[DM94] Marcello D'Agostino and Marco Mondadori. The taming of the cut. *Journal of Logic and Computation*, 4:285–319, 1994.

[Dôs93] Kosta Dôsen. A historical introduction to substructural logics. In P. Schroeder Heister and Kosta Dôsen, editors, *Substructural Logics*, pages 1–31. Oxford University Press, 1993.

[Gab96] Dov M. Gabbay. *Labelled Deductive Systems, Volume 1 - Foundations.* Oxford University Press, 1996.

[Gen35] Gerhard Gentzen. Unstersuchungen über das logische Schliessen. *Math. Zeitschrift*, 39:176–210, 1935. English translation in [Sza69].

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Rus96] Alessandra Russo. Generalising propositional modal logic using labelled deductive systems. In *Applied Logic Series (APLS), "Frontiers of Combining Systems, First International Workshop"*, volume 3, pages 57–73, 1996.

[Sti81] M. Stickel. A Unification Algorithm for Associative-Commutative Functions. *Journal of the ACM*, 28(3), 1981.

[Sza69] M. Szabo, editor. *The Collected Papers of Gerhard Gentzen*. North-Holland, Amsterdam, 1969.