HAIL: Hybrid Abductive-Inductive Learning

Oliver Ray
or@doc.ic.ac.uk

Technical Report: 2003/6

Department of Computing
Imperial College London
180 Queen's Gate
London SW7 2BZ

Abstract

The learning system Progol5 and the underlying inference method of Bottom Generalisation are firmly established within Inductive Logic Programming (ILP). But despite their success, it is known that Bottom Generalisation, and therefore Progol5, are restricted to finding hypotheses that lie within the semantics of Plotkin's relative subsumption. This report provides a rational reconstruction, critical analysis, and extension of Progol5 and Bottom Generalisation. In particular, a previously unsuspected incompleteness of Progol5 with respect to Bottom Generalisation is exposed, and a new approach is proposed, called *Hybrid Abductive Inductive Learning* (HAIL), that integrates the ILP principles of Progol5 with Abductive Logic Programming (ALP). A proof procedure is described, also called HAIL, that not only overcomes this newly discovered incompleteness of Progol5, but further generalises Progol5 by computing multiple clauses outside Plotkin's relative subsumption. A semantics is presented, called *Kernel Generalisation*, which extends that of Bottom Generalisation and includes the hypotheses constructed by HAIL.

Contents

Chapter 1

Introduction

This report is concerned with two areas of Artificial Intelligence: Inductive and Abductive Logic Programming. In particular, this report considers two prominent systems – Muggleton's inductive system Progol, and the abductive-SLD procedure of Kakas and Mancarella – and proposes a novel approach, called Hybrid Abductive Inductive Learning, for their generalisation and integration. This chapter begins in Section 1.1 by introducing the key background concepts and describing the principal motivations of this work. Section 1.2 then outlines the main objectives of this research and provides a hypothesis statement. This is followed in Section 1.3 with a summary of the contributions of so far achieved. Finally, Section 1.4 outlines the structure of this report.

1.1 Motivation

Machine Learning is the branch of Artificial Intelligence that seeks to better understand and deploy learning systems through the analysis and synthesis of analogous processes in machines. The specific task of generalising from positive and negative examples relative to given background knowledge has been much studied in Machine Learning, and when combined with a first-order clausal representation is known as *Inductive Logic Programming* (ILP) [Mug91, MR94]. Given background knowledge B, positive examples E^+, and negative examples E^-, the task of ILP is to find a hypothesis H that – when added to B – entails E^+ and is consistent with E^-. In practice, this hypothesis H is constructed from among several possible hypotheses by some particular method of inductive inference, using some form of user-specified language and search bias.

The *Progol* system of Muggleton [Mug95] is a widely-applied ILP system that has been successfully applied in numerous real world applications; most notably in the field of bioinformatics (see for example [SMSK96, FMPS98, TMS01]). The Progol language bias is determined by a set of user supplied mode declarations that define a hypothesis space within which hypothesised clauses must fall. The Progol search bias is determined by an in-built preference metric, called *compression*, that prefers shorter hypotheses over longer hypotheses. Underlying Progol is the inference method of *Bottom Generalisation* [Mug95, Yam99], which given background knowledge B and a single positive seed example e, constructs and generalises a clause, denoted $Bot(B, e)$ and called the *Bottom Set [Mug95] of B and e*, to return a hypothesis h that when added to B entails e.

Despite the success of Progol, it is shown in [Yam97, Yam99] that Bottom Generalisation, and therefore Progol, are limited to deriving clauses h that *subsume e relative to B* (in the sense of [Plo71]). In addition to this semantic restriction, Progol is also subject to several sources of procedural incompleteness. It is true that some of this incompleteness has been deliberately introduced into Progol in order to improve efficiency. The restriction to Horn clause logic, for example, allows for the use of efficient logic programming technologies, while retaining sufficient expressivity for many applications. But much of the incompleteness of Progol is less intentional. For example, until recently, Progol suffered from a common limitation called *Observation Predicate Learning* (OPL) [MB00], whereby h is restricted to defining the same predicate as e, and which was found to hinder the use of Progol in real applications.

Progol5 [MB00] is the most recent system in the Progol family. Like its predecessors Progol5 is based on an approach called *Mode Directed Inverse Entailment* (MDIE) [Mug95], which refers to the extensive use made of mode-declarations in both the construction and generalisation of the Bottom Set. But in addition Progol5 incorporates a significant extension of MDIE, called *Theory Completion with Inverse Entailment* (TCIE) [MB00], that partially overcomes the limitation of OPL. This enhancement is due to the introduction of a routine, called STARTSET, for computing non-trivial atoms in the head of the Bottom Set. But although it improves substantially upon Progol4, there remain several significant limitations from which Progol5 still suffers – some of which are addressed in this report. Interestingly, it will be revealed that one particularly unfortunate such limitation is due to the inherent incompleteness of the STARTSET algorithm itself.

It turns out that the computation of atoms in the head of the Bottom Set and the operation of the STARTSET routine are intimately connected to *Abductive Logic Programming* (ALP) [KKT92], which is a close relative of ILP. The field of ALP is a branch of Artificial Intelligence that is concerned with reasoning under incomplete information in a logic programming context. ALP automates the task of explaining a goal relative to given theory and integrity constraints. Thus given theory T, goal g, and integrity constraints IC, the task of ALP is to find a hypothesis Δ that together with T entails g and is consistent with IC. But whereas ILP seeks new *clauses* H to generalise given *clauses* E^+ and E^-, ALP is less ambitious, seeking instead new *facts* Δ to explain given *facts* g. In addition, the hypothesis Δ is further restricted to a set of so-called abducibles, which are a set of predefined ground atoms that define the allowable assumptions. Prominent within ALP is the Abductive-SLD (ASLD) proof procedure of Kakas and Mancarella [KM90], which augments conventional SLD resolution with the facility for abductive reasoning.

The connection between STARTSET, Bottom Generalisation, and ALP is established in [MB00], where it is suggested that STARTSET can be viewed as a form of abduction, and in [Yam00], where it is shown that atoms in the head of the Bottom Set can be computed by an abductive proof procedure. However, so far, neither the relationship between STARTSET and Bottom Generalisation, nor the relationship between Bottom Generalisation and ALP, have been sufficiently well analysed. The latter has been formally studied only in the context of definite clause logic, and the former has not been formally studied at all. The need for such a study is increased by the discovery, made in this report, of a previously unsuspected incompleteness of the STARTSET routine with respect to its intended semantics.

This newly discovered incompleteness of STARTSET, and consequently of Progol5 as a whole, is illustrated in Example (Ex.1) below. In this example, p, q, r and s are propositions and the background knowledge, denoted B_1, is the clausal theory consisting of the three clauses (in standard Prolog notation) p :- q, s and q :- r and s :- r. The seed example e_1 is the ground atomic clause p. It is shown in this report that while the hypothesis h_1 consisting of the atomic clause r is derivable by Bottom Generalisation, it cannot be computed by Progol5. This surprising result indicates that a deeper analysis is required of Progol5, and in particular of the soundness and incompleteness of the STARTSET algorithm. It also raises several questions concerning the origin, nature, and extent of this incompleteness, as well as motivating the investigation of techniques capable of surmounting it.

$$B_1 = \{p \text{ :- } q, s\} \cup \{q \text{ :- } r\} \cup \{s \text{ :- } r\} \qquad e_1 = p \qquad h_1 = r \qquad \text{(Ex.1)}$$

(Ex.1) is not just a trivial example that a reasonable ILP system might be expected to solve; it is also an exemplar of general pattern of reasoning likely to prove useful in practice. This pattern can be described as follows: some desired effect (the atom p, here) is explained by a number of preconditions (q and s) that share a common cause (r), and the hypothesis is found by identifying this common cause (r). A related pattern of reasoning, which is exemplified in (Ex.2) below, can be described as follows: some desired effect (p) is the result of a certain property (the unary predicate t) holding of some class of individuals (a and b), and the hypothesis is found by extending this property to a wider class of individuals (all X).

$$B_2 = \{p \text{ :- } t(a), t(b)\} \qquad e_2 = p \qquad h_2 = t(X) \qquad \text{(Ex.2)}$$

The relationship between (Ex.1) and (Ex.2) can be seen by informally identifying proposition q with the atom $t(a)$, proposition s with the atom $t(b)$, and proposition r with the formula $\forall X(t(X))$. The first clause of B_1 becomes identical to B_2, and the remaining two clauses of B_1 are tautologies. It is interesting to note that these two patterns are typical examples of what are sometimes referred to in the literature as 'explanatory' and 'predictive' induction, or 'abductive' and 'inductive' reasoning [FK00]. Note also that many much more interesting examples are easily obtained as relatively straightforward generalisations and combinations of these basic patterns[1].

Yet despite its apparent simplicity, neither can (Ex.2) be solved by Progol5. For it is shown in this report that h_2 is not derivable by Bottom Generalisation from B_2 and e_2, and therefore it cannot be computed by Progol5. But, so far, this and many other inherent limitations of Bottom Generalisation have not been satisfactorily addressed. Although the discovery in [Yam97] of the semantic incompleteness of Bottom Generalisation did prompt the development of a number of complete hypothesis finding schemes for full clausal logic, such as [YF00, Ino01a], where such systems have been implemented, they are highly computationally expensive and only semi-automatic. Consequently, these methods have not yet approached the same degree of practical success as Progol.

[1]For example, the background theory $B = \{phd(X) \text{ :- } pass(supervisor, X), pass(examiner, X)\}$ - stating that a student earns a PhD if his supervisor awards him a pass, and so does his examiner - is a generalisation of (Ex.2): obtained by identifying i) p and t with phd and $pass$ (adding an additional argument), and ii) a and b with $supervisor$ and $examiner$. Given the example $e = phd(albert)$ the hypothesis $h = pass(X, albert)$ is obtained directly from the pattern above. Clearly one can generalise further; for example by making $supervisor$ and $examiner$ more realistically into unary functions. Alternatively, the background theory $B = \{phd(X) \text{ :- } pass(supervisor, X), pass(examiner, X)\} \cup \{pass(examiner, X) \text{ :- } pass(supervisor, X)\}$ - stating in addition that if the supervisor awards a pass then the examiner will follow suit - is a special case of (Ex.1): obtained by identifying i) p with $phd(X)$, ii) q with $pass(supervisor, X)$, and iii) r and s with $pass(examiner, X)$; and by dropping the resulting tautology. Given the example $e = phd(albert)$ the hypothesis $h = pass(supervisor, albert)$ is obtained from the pattern above.

Another fundamental limitation of Bottom Generalisation is that it can hypothesise only one single clause in response to each seed example. This restriction can often result in Progol5 failing to find the best hypothesis, and sometimes in failing to find any hypothesis at all. For instance, given B_3 and e_3 as in Example (Ex.3) below, the only hypothesis that can be derived by Bottom Generalisation is e_3 itself. The more likely hypothesis H_3, consisting of the *two* ground atomic clauses q and r, cannot be derived by Bottom Generalisation. An attempt to overcome this limitation is described in [Yam00], where a method based on Bottom Generalisation is proposed that is able to hypothesise a set of unit clauses in response to a single seed example. But while that method would be able to solve Example (Ex.3), it cannot be applied to *Horn* clauses, and it cannot hypothesise *multiple* non-unit clauses.

$$B_3 = \{p \text{:-}\, q, r\} \qquad e_3 = p \qquad H_3 = \{q\} \cup \{r\} \qquad \text{(Ex.3)}$$

In conclusion, it is believed this discussion supports the belief that it would comprise a significant and useful achievement if a natural and efficient generalisation of Progol5 and Bottom Generalisation were to be obtained that supports, in Horn clause logic, the three reasoning patterns exemplified in (Ex.1), (Ex.2) and (Ex.3) above. One promising means to this end, which is identified and pursued in this report, is inspired by a closer examination of the relationship between Bottom Generalisation and ALP, and by combining abductive and inductive learning within a cycle that generalise Progol5. For, until now, the integration of Bottom Generalisation and ALP has not been studied in sufficient generality, and existing work has used abduction to compute just one atom at a time, for use in the head of a single induced clause. A natural extension is to employ ALP in its usual setting of computing multiple atoms and to use those atoms in the heads of several simultaneously induced clauses.

1.2 Objective

The goal of this research is to develop a new ILP approach that builds on the practical success of Progol and Bottom Generalisation by exploiting their many proven strengths while overcoming some of their limitations. What is envisioned is a successor to Progol5 that operates on precisely the same inputs, in much the same way as Progol5, but overcomes some of the procedural and semantic incompletnesses outlined above. The hypothesis is that this vision can be realised by integrating the full ASLD procedure within an inductive cycle of learning that generalises Progol5. In particular, by carefully exploiting multiple atom abductive hypotheses it is believed that practical procedures can be developed that will be able to infer multiple clauses in response to a single seed example, and to derive useful hypotheses outside the semantics of Bottom Generalisation.

There are two key objectives. The first objective is to provide a fuller analysis of the relationship between Progol5, Bottom Generalisation, and ALP. This requires a formal investigation of the soundness and completeness of the Progol5 STARTSET with respect to Bottom Generalisation, a mathematical characterisation of any incompleteness thereby uncovered, and a formal investigation of the relationship between Bottom Generalisation and ALP within Horn Clause logic.

The second objective is to generalise the Progol approach. A proof procedure is required that behaves much like Progol5, but is able to infer multiple clauses from a single seed example, and supports the three reasoning patterns exemplified in the previous section. A semantics is required that generalises Bottom Generalisation, and with respect to which the soundness and completeness of the new procedure can be established. Finally, in order to validate the approach, a prototype should be implemented and applied to a small case study.

1.3 Contribution

This report provides a rational reconstruction, critical analysis, and extension of Progol5 and Bottom Generalisation. The relationship between Bottom Generalisation and relative subsumption is investigated, and the clear and intuitive role played by the Bottom Set in the theory of relative subsumption is established. Bottom Generalisation is motivated simply and naturally within the context of the well-known ILP cover-set approach. The Progol5 algorithm and in particular the STARTSET routine is analysed, confirming its suspected soundness, but exposing a previously unsuspected incompleteness with respect to Bottom Generalisation. Two classes of literals are identified that STARTSET fails to compute. The notion of *vacuous literal* is introduced and used to characterise the first incompleteness class. It is argued, however, that such literals are of little practical value, and that their non-computation by STARTSET is in fact beneficial to Progol5. The notion of *c*-refutation* is introduced in order to characterise the second incompleteness class. This time the failure of STARTSET is serious, resulting in the inability of Progol5 to construct many hypotheses that would be expected in practice. The connection between ALP and Bottom Generalisation is investigated in the context of Horn clause logic, and a reformulation of the Bottom Set is obtained that reveals two distinct abductive and deductive components. A novel approach is proposed, called *Hybrid Abductive Inductive Learning* (HAIL), that integrates abductive, deductive, and inductive reasoning within a cycle of learning that generalises Progol5. Unlike previous approaches for integrating abduction and Bottom Generalisation, this new approach exploits the ability of ALP to compute multiple atom hypotheses in order to induce multiple clauses. A proof procedure is proposed, also called HAIL, that not only overcomes this newly discovered incompleteness of Progol5, but further generalises Progol5 by computing multiple clauses in response to a single seed example and deriving hypotheses outside Plotkin's relative subsumption. A semantics is presented, called *Kernel Generalisation*, which extends that of Bottom Generalisation and includes the hypotheses constructed by the HAIL proof procedure. A refinement of this semantics is described, called Kernel Set Subsumption, that is more amenable to logic programming methods. The key concept, called a *kernel*, is a logical formula that is a generalisation of the Bottom Set. The notion of K-refutation is introduced as a first attempt to characterise the class of hypotheses derivable by HAIL.

1.4 Structure

This report is structured as follows. Chapter 2 reviews the basic notation and terminology used in this report. Chapter 3 provides an introduction to ILP followed by a rational reconstruction and critical analysis of Progol5. Chapter 4 provides a brief introduction to ALP and overview of the ASLD proof procedure. Chapter 5 introduces Hybrid Abductive Inductive Learning. This chapter introduces the semantics of *Kernel Generalisation*, the refinement of *Kernel Set Subsumption*, and the HAIL proof procedure. Chapter 6 compares this approach with related work. Finally, the report concludes with a summary and a discussion of future work.

Chapter 2

Preliminaries

This report assumes a familiarity with classical first order logic, clausal form logic, and logic programming. The purpose of this chapter is to briefly review the relevant background concepts and to summarise the basic notation and terminology used throughout this report. First, Section 2.1 recalls some standard definitions and results from logic and logic programming. Then, Sections 2.2 and 2.3 define and discuss the key notions of Skolemisation, incapitation, relative subsumption and c-derivations. Finally, Section 2.4 specifies the knowledge base of a simple running example that will be used throughout this report.

2.1 Notation and Terminology

This section recalls some standard definitions and results from the fields of classical logic and logic programming. The following paragraphs provide a brief summary whose main purpose is that of introducing the basic notation and terminology used in this report. For further details the reader is referred to the references provided.

First Order Logic. The following treatment closely follows [End72]. A *term* is defined recursively as a constant symbol c, a variable symbol X or a function symbol f (of arity $m \geq 1$) applied to an m-tuple of terms. An *atom* is a proposition symbol q, or a predicate symbol p (of arity $n \geq 1$) applied to an n-tuple of terms. The notation x/k denotes that x has arity k, where x represents a function or predicate symbol and k is a positive integer. If $A = p(t_1, \ldots, t_n)$ is an atom and p^* is a predicate of arity n, then the notation A^* will denote the atom $p^*(t_1, \ldots, t_n)$. A *literal* L is an atom A or the negation of an atom $\neg A$. The functions $pred(L)$ and $var(L)$ return respectively the predicate symbol, and the set of all variables appearing in L. The notation $\overline{L}$ denotes the *complement* of L and is obtained by negating L if is un-negated, and un-negating L if L is negated. Formulae of classical logic will be written with the standard *connectives* $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$, the *quantifiers* $\forall, \exists$, and the logical *constants* $\top, \bot$. The relations $\models$, $\equiv$ and $\doteq$ will denote classical logical entailment, logical equivalence, and syntactic identity. The relation $\models$ will also be referred to as *logical implication.* Let F, G and H denote arbitrary logical formulae. If $G \models H$, then G is said to *entail* or to *logically imply* H. Given H, the task of finding some G such that $G \models H$ is called the task of *straight inverse implication* or just *inverse implication.* If $F \wedge G \models H$, then G is said to *logically imply* H *relative* to F. Following [NCdW97], this will sometimes be written $G \models_F H$. Given F and H, the task of finding some G such that $F \wedge G \models H$ is called the task of *relative inverse implication.* The metalogical equivalences $F \wedge G \models H$ iff $F \models G \rightarrow H$ and $F \wedge G \models \bot$ iff $F \models \neg G$ will be referred to as the *entailment theorem*, and the equivalence $F \wedge G \models H$ iff $F \wedge \neg H \models \neg G$ will be referred to as the *principle of Inverse Entailment* (IE). The conjunction and disjunction operators are generalised to operate over sets of formulae, so that if $\mathcal{F} = \{F_1, \ldots, F_n\}$ is a set of formulae then $\bigwedge \mathcal{F}$ is the formula $F_1 \wedge \ldots \wedge F_n$, and $\bigvee \mathcal{F}$ is the formula $F_1 \vee \ldots \vee F_n$. By convention $\bigwedge \emptyset = \top$ and $\bigvee \emptyset = \bot$. A *logical theory* is an implicitly conjoined set of formulae. Let T and L be logical theories. A *conservative extension* of T *with respect to* L is any theory S such that $S \models \phi$ iff $T \models \phi$ for all formulae $\phi \in L$. When used in this way, L will be referred to as the '*language*' of T. A substitution $\theta = \{V_1/t_1, \ldots V_n/t_n\}$ is a set of bindings V_i/t_i, where each V_i is a *distinct* variable and each t_i is a term. Note that it *is not* assumed that V_i does not occur in t_j for all i, j – thus *idempotency* of substitutions is not enforced. The *application* of a substitution θ to a term t is denoted $t\theta$ and defined as the term resulting from the replacement of each occurrence of the variable V_i in t by the term t_i. The *application* of a substitution θ to a formula F is denoted $F\theta$ and defined as the formula resulting from the replacement of each *free* occurrence of the variable V_i in F by the term t_i. Note that it is assumed, if necessary by a suitable renaming of bound variables in F, that for all i there is no variable $U_i \neq V_i$ occurring in t_i such that V_i occurs free in F within the scope of a quantifier that binds U_i – thus *variable capture* is not permitted. Given a formula F, then a substitution $\sigma = \{V_1/c_1, \ldots V_n/c_n\}$ is a *Skolemising substitution* for F iff is σ is a substitution that binds distinct variables V_i to distinct constants c_i not appearing in F. If σ is a Skolemising substitution then σ^{-1} will denote the *anti-Skolemisation* of σ. When σ^{-1} is applied to a term or a formula, it behaves as a substitution, but replaces each Skolem constant c_i by the corresponding variable V_i. The sets $\mathcal{GA}$ and $\mathcal{GL}$ will denote respectively the sets of ground atoms and literals of an assumed first order language $\mathcal{L}$ with signature Σ.

Clausal Form Logic. The following treatment closely follows [Rob65] and [Llo87]. A *clause* $C = \{A_1, \ldots, A_m, \neg B_1, \ldots, \neg B_n\}$ is a set of literals, the A_i and B_j being atoms. Any clause C appearing in a logical formula is understood to denote the disjunction of its literals, and *unless* indicated otherwise, all variables in C are assumed to be universally quantified at the front of the clause. A clause C will often be written in the implicative form $A_1, \ldots, A_m \text{:-} B_1, \ldots, B_n$. The *sets* $C^+ = \bigcup_i A_i$ and $C^- = \bigcup_j B_j$ are called respectively the *head* and *body* of C. If C is a clause and σ is a substitution, then the *application* of σ to C, denoted C_σ is the clause $\{A_1\sigma, \ldots, A_m\sigma, \neg B_1\sigma, \ldots, \neg B_n\sigma\}$. For convenience, C_σ will sometimes be written $C\sigma$; it being understood that the explicit substitution is to be performed prior to the implicit quantification. As is the case with formulae, so a substitution σ is called a *Skolemising substitution* for a clause C iff σ binds each distinct variable in C to a *fresh* Skolem constant. A *Horn clause* is a clause having at most one atom in the head. A non-empty Horn clause is either: a *fact* A, a *rule* $A \text{:-} B_1, \ldots, B_n$, or a *denial* $\text{:-} B_1, \ldots, B_n$. The *empty-clause* is represented $\Box$ and denotes the logical constant $\bot$. A clause with exactly one head atom is called *definite*, and a clause with none (i.e. a denial) is called *negative.* A *clausal theory* or *theory* T is a set of clauses $\{C_1, \ldots, C_n\}$. Any theory T appearing in a logical formula is understood to denote the conjunction of its clauses. A theory composed entirely of Horn/definite/negative clauses is called a *Horn/Definite/Integrity Theory.* The theory $\hat{C}$ denotes that obtained from C by standardising apart all of the clauses (so that no two mention the same variable). The *empty-theory* is represented with the symbol $\blacksquare$ and denotes the logical constant $\top$. For any clause $C = A_1, \ldots, A_m \text{:-} B_1, \ldots, B_n$, the *clausal complement* of C is the clausal theory, denoted $\overline{C}$, consisting of the clauses $\text{:-} A_1\sigma$ and $\ldots$ and $\text{:-} A_m\sigma$ and $B_1\sigma$ and $\ldots$ and $B_n\sigma$, where σ is a Skolemising substitution for C. The extended notation $\overline{C_\sigma}$ will be used whenever it is necessary to refer to the particular Skolemising substitution. Given three clauses C_1, C_2 and C_3, the notation $C_1 \in \Re(C_2, C_3)$ denotes that C_1 is a *resolvent* of C_2 and C_3. The terminology *input clause* is used to denote a leaf in a resolution tree. Note the following relationships, which follow from the above definitions. First, $C \equiv \forall(A_1 \vee \ldots \vee A_m \vee \neg B_1 \vee \ldots \vee \neg B_n) \equiv \forall(A_1 \vee \ldots \vee A_m \leftarrow B_1 \wedge \ldots \wedge B_n)$. Second, $C \equiv \forall(\bigwedge C^- \rightarrow \bigvee C^+)$. Third, $\text{:-} B_1, \ldots, B_n \equiv \neg\exists(B_1 \wedge \ldots \wedge B_n)$. Fourth, $T \equiv C_1 \wedge \ldots \wedge C_n$. Fifth, $\overline{C_\sigma} = \{\{\neg A_1\sigma\}, \ldots, \{B_n\sigma\}\} \equiv \neg A_1\sigma \wedge \ldots \wedge B_n\sigma \equiv \neg C_\sigma$.

Definite Logic Programming. A definite theory P is also called a Definite Logic Program. A *query* $Q = \{Q_1, \ldots, Q_n\}$ is a finite set of atoms (referred to as *calls*). For convenience a query Q will be written in the interrogative form $?Q_1, \ldots, Q_n$. A query Q is said to *succeed* from a program P *with answer substitution* ϕ iff there is a SLD derivation of $\Box$ from $P \cup \{\text{:-} Q_1, \ldots, Q_n\}$, and ϕ is the composition of the unifiers, restricted to the variables in Q. This eventuality will be written $\phi \in \mathcal{S}\text{LD}(P, Q)$, or more compactly $P \vdash Q\phi$. The application of ϕ to Q will be denoted $Q\phi$ and called an *answer.* Wherever an answer $Q\phi$ appears in a logical formula, it is understood to denote the universal closure of the conjunction of the calls of Q, to which the substitution ϕ has first been applied. Note that $Q\phi \equiv \forall(Q_1\phi \wedge \ldots \wedge Q_n\phi)$. The simplified notations $\mathcal{S}\text{LD}(P, Q)$ and $P \vdash Q$ will be used when it is not necessary to refer to a particular answer substitution. The following soundness and completeness results are well known: If $P \vdash Q\phi$ then $P \models Q\phi$. If $P \models Q\phi$ then there exist substitutions ϕ' and ϕ'' such that $P \vdash Q\phi'$ and $\phi = \phi'\phi''$. More generally, if H is a Horn clause theory then $H \vdash Q\phi$ implies $H \models Q\phi$, and $H \models Q\phi$ implies that either $H \models \Box$ or there exist substitutions ϕ' and ϕ'' such that $H \vdash Q\phi'$ and $\phi = \phi'\phi''$. The completeness results just stated are instances of the *subsumption theorem* [NCdW97], and will be referred to as such.

Theta-Subsumption. A clause C is said to *θ-subsume* a clause D, written $C \succeq D$, iff $C\theta \subseteq D$ for some substitution θ. A clause is *reduced* iff it does not θ-subsume some proper subset of itself. The relation $\succeq$ induces a *lattice* ordering on the set of reduced clauses (up to renaming of variables). The *bottom* element in this ordering is the empty-clause $\Box$ and represents a logical contradiction, and the *top* element is the clause in which all atoms of the Herbrand base appear in both head body and represents a logical tautology. By a slight abuse of notation, this top element will be denoted $\blacksquare$ and called the *null-clause.* By a further abuse of notation, both the empty-clause and the null-clause will sometimes be written $\emptyset$. The θ-subsumption *sub-lattice* bounded from below by a clause C and from above by a clause D will be denoted $[C, D]$, and the entire θ-subsumption lattice is therefore denoted $[\Box, \blacksquare]$. It is true that if $C \succeq D$ then $C \models D$, but the converse does not hold in general: for example, let $C = n(s(X)) \text{:-} n(X)$ and $D = n(s(s(X))) \text{:-} n(X)$. The notion of θ-subsumption is generalised to operate between clausal theories as follows: A clausal theory S θ-subsumes a theory T, written $S \sqsupseteq T$, iff every clause in T is θ-subsumed by at least one clause in S.

Normal Logic Programming. A NAF-literal is an atom A or the *negation-as-failure* of an atom $\backsim A$. A *Normal Clause* is written $A \text{:-} B_1, \ldots, B_m, \backsim C_1, \ldots, \backsim C_n$ where the A and B_i and C_j are atoms. In other words, a normal clause consists of a single atom in the head, and a set of NAF literals in the body. A set of normal clauses is called a *Normal Logic Program.* A query $Q = ?Q_1, \ldots, Q_n$ *succeeds* from a normal logic program N with *answer substitution* ϕ iff there is a SLDNF derivation of $\Box$ from $N \cup \{:\!\text{-}\, Q_1, \ldots, Q_n\}$ and ϕ is the composition of the unifiers, restricted to the variables in Q. This eventuality will be written $\phi \in S\text{LDNF}(N, Q)$ or $N|\!\sim Q\phi$. The following soundness result is well known: If $N|\!\sim Q\phi$ then $Comp(N) \models Q\phi$ where $Comp(N)$ denotes the *Clark completion* [Cla78] of N. There is no universally accepted logical semantics for normal logic programs; instead there are a number of alternative *canonical model* or *preferred model* constructions, which include the *stable*, *perfect* and *well-founded* models [GL88, Prz89, GRS91]. If $\mathcal{M}$ is a canonical model construction and N is a normal logic program, then $\mathcal{M}_N$ will denote *canonical model* of N (under $\mathcal{M}$). Any logical formula f satisfied by the canonical model $\mathcal{M}_N$ will be called a *consequence* of N (under $\mathcal{M}$), and this will be written $N \models_{\mathcal{M}} f$. Note that if N is a definite logic program then all of the semantics mentioned above coincide with the Least Herbrand Model (LHM) of N.

Logic Programming Restrictions. A normal clause $A :\!\text{-}\, L_1, \ldots, L_k$ is *constrained* iff $var(A) \supseteq var(L_i)$ for all $1 \leq i \leq k$. A normal logic program N is constrained iff all of its clauses are constrained. A denial $:\!\text{-}\, L_1, \ldots, L_k$ is *constrained with respect to* atom A iff the clause $A :\!\text{-}\, L_1, \ldots, L_k$ is constrained. A normal logic program N is *stratified* [ABW87] iff there exists a mapping $|\;|$ from $pred(N)$ to the natural numbers $\mathcal{N}$, such that if $A :\!\text{-}\, B_1, \ldots, B_m, \backsim C_1, \ldots, \backsim C_n$ is any clause in N then: i) $|pred(A)| \geq |pred(B_i)|$ for all $1 \leq i \leq m$, and ii) $|pred(A)| > |pred(C_j)|$ for all $1 \leq j \leq n$. A normal logic program N is *locally stratified* iff there exists a mapping $|\;|$ from the Herbrand base $B(N)$ to the natural numbers $\mathcal{N}$, such that if $a :\!\text{-}\, b_1, \ldots, b_m, \backsim c_1, \ldots, \backsim c_n$ is any ground instance of a clause in N then: i) $|a| \geq |b_i|$ for all $1 \leq i \leq m$, and ii) $|a| > |c_j|$ for all $1 \leq j \leq n$. The definitions of *hierarchical* and *acyclic* programs, are obtained by replacing the relation $\geq$ by the relation $>$ in conditions i) above for stratified and locally stratified programs respectively.

2.2 Skolemisation and Incapitation

Having informally reviewed in the previous section the basic notation and terminology used in this report, this section considers more formally two further concepts, called Skolemisation and incapitation, that will be used in the sequel as a basis for reasoning about two important standardising ILP transformations. First, recall from classical logic the definition of Skolemisation, shown in Definition 1 below.

Definition 1 (Skolemisation). The *Skolemisation* of a formula F is a formula, denoted $S(F)$, obtained by converting F into a prenex normal form F', and then removing from F' each existential quantifier $\exists X$ and replacing every occurrence of X previously bound to that quantifier by a term $f_X(Y_1, \ldots, Y_n)$, where f_X is a *fresh* Skolem symbol, and the $Y_1, \ldots, Y_n$ with $n \geq 0$, are all those variables within the scope of a universal quantifier at the point in F' where the existential quantifier previously occurred. ♦

The extended notation $S_k(F)$ will be used whenever it is necessary to refer to the set $k = \{f_X^1, \ldots, f_X^n\}$ of Skolem symbols newly introduced in $S(F)$, and the even stronger form $S_\sigma(F)$ will be used when each variable symbol in F is bound by at most one existential quantifier, and it is necessary to refer to the substitution $\sigma = \{X_1/t_1, \ldots, X_n/t_n\}$ that replaces each existentially quantified variable X_i by a unique Skolem term t_i.

Note that for any formula F it holds that: i) $S(F) \models F$, and ii) $S(F) \models \bot$ iff $F \models \bot$. If C is a clause then $S(C) \equiv C$. If $\neg C$ is the negation of a clause then $S_\sigma(\neg C) \equiv \neg(C_\sigma)$ which by definition of clausal complements can be written $\overline{C_\sigma}$ since σ is a Skolemising substitution for C. If T is a clausal theory then $S(T) \equiv T$. And if T is a theory and C is a clause, then $S_\sigma(T \wedge \neg C) \equiv T \cup \overline{C_\sigma}$. Note also that $S_k(F)$ is a conservative extension of F with respect to any formulae not containing any Skolem symbol in k. Thus, given a clausal theory T, a clause C, and a formula G it follows that $T \wedge \overline{C} \models G$ iff $T \wedge \neg C \models G$, providing that G and T contain no Skolem constant introduced in $\overline{C}$.

The properties of Skolemisation described above will be used in the next chapter to reason about a standardisation procedure that involves the replacement of variables by previously unused constants. Another process used in this standardisation procedure is the insertion of a previously unused proposition into the heads of negative clauses. This latter process will henceforth be referred to as *incapitation* (meaning literally 'insertion into the head of') and the special symbol ff will from now on be used to denote this proposition symbol, as shown in Definition 2 below.

Definition 2 (Incapitation). If C is a Horn clause, and if ff is a proposition not appearing in C, then the *incapitation* of C (with respect to ff) is the definite clause denoted $C_{\vec{\mathsf{ff}}}$ obtained by adding ff into the head of C, if C is a negative clause, and leaving C unchanged otherwise. If T is a Horn clause theory, then $T_{\vec{\mathsf{ff}}}$ is the theory obtained by incapitating each individual Horn clause in T. ♦

Note that if D is a definite theory then $D_{\vec{\mathsf{ff}}} = D$. If, in addition, T is a Horn theory, then $(D \cup T)_{\vec{\mathsf{ff}}} = D \cup T_{\vec{\mathsf{ff}}}$, which is a definite theory. Note also that the Horn theory $T_{\vec{\mathsf{ff}}} \cup \{ \text{:-}\, \mathsf{ff}\}$ is a conservative extension of T with respect to any formulae not containing the proposition ff. Thus, given a clausal theory T, a clause C, and a formula G it follows that $T_{\vec{\mathsf{ff}}} \cup \{ \text{:-}\, \mathsf{ff}\} \models G$ iff $T \models G$, providing that G does not contain the proposition ff. Moreover, by letting $G = \bot$ and noting that this atom does not contain the proposition ff, then by the entailment theorem $T_{\vec{\mathsf{ff}}} \models \mathsf{ff}$ iff $T \models \Box$, where the proposition ff is assumed not to occur in T.

2.3 Relative Subsumption and C-Derivations

It was mentioned in the previous chapter that the notions of relative subsumption and c-derivations characterise the hypotheses derivable by the ILP inference method Bottom Generalisation, which plays a central role in this report. This section recalls the definitions of these two concepts and considers the relationship that exists between them. The reader is advised that while the definitions are based on [Plo71], they are subject to notational revisions introduced by [Yam97] and [NCdW97].

Definition 3 (C-Derivation). A *c-derivation* of a clause D from a clausal theory T *with respect to* a clause C, is a resolution derivation of the clause D from the clauses $T \cup \{C\}$, in which C is used at most once as an input clause (i.e. a leaf). A c-derivation of the empty-clause is called a *c-refutation* The existence of a c-derivation will be denoted $(T, C) \vdash_c D$. ♦

Definition 4 (Relative Subsumption). A clause C *subsumes* a clause D *relative* to a theory T, written $C \succeq_T D$, iff the following three equivalent conditions are satisfied:

i. there exists a substitution ϕ such that $T \models \forall(C\phi \to D)$

ii. there exists a clause F such that $T \models \forall(D \leftrightarrow F)$ and $C \succeq F$

iii. there exists a clause E such that $D \equiv \top$ or $(T, C) \vdash_c E$ and $E \succeq D$ ♦

The close relationship between relative subsumption and c-derivations is evidenced in condition iii) above. Following [Yam99], this relationship can be more conveniently expressed in terms of c-refutations as $C \succeq_T D$ iff $T \models D$ or $(T \cup \overline{D}, C) \vdash_c \Box$. Moreover, it is now shown in Proposition 2.3.2 below, that this relationship can be expressed even more simply by dropping the unnecessary disjunct $T \models D$. But first Lemma 2.3.1 formalises the relationship between c-refutations and the simpler notion of θ-subsumption.

Lemma 2.3.1 (Deciding θ-Subsumption via C-Refutation[1]). *Given any clauses C and D. As usual, let $\overline{D}$ denote the clausal complement of D, where none of the Skolem constants newly introduced in $\overline{D}$ occur in C or D. Then*

$$(\overline{D}, C) \vdash_c \Box \quad \textit{iff} \quad C \succeq D \textit{ or } D \equiv \top$$

Proof. [**If-Part**]. **Case 1**: If $C \succeq D$ then $C\theta \subseteq D$ by definition of θ-subsumption. Writing D explicitly as the set of literals $\{d_1, \ldots, d_n\}$, observe that $\overline{D}$ is the set of ground unit clauses $\{\{\overline{d_1\sigma}\}, \ldots, \{\overline{d_n\sigma}\}\}$, where σ is a Skolemising substitution. Then $C\theta\sigma \subseteq D\sigma$ (c.f. Lemma A.7.2.1). Now construct a linear resolution derivation $C\theta\sigma, \ldots, \Box$ by successively resolving away literals from the descendents of $C\theta\sigma$ with the complementary clause in $\overline{D}$ (which clearly exists). By the lifting lemma, there is a corresponding derivation $C, \ldots, \Box$, and since C is only used once as an input clause it follows that $(\overline{D}, C) \vdash_c \Box$ by Definition 3. **Case 2**: If $D \equiv \top$ then $\overline{D}$ must contain some ground atom A and its complement $\neg A$. Hence $\Box$ is trivially derivable from $\overline{D}$ without using C at all. Therefore $(\overline{D}, C) \vdash_c \Box$ by Definition 3.

[**Only-If-Part**]. If $(\overline{D}, C) \vdash_c \Box$ then either C was used once in the derivation or it was not used at all. Writing $D = \{d_1, \ldots, d_n\}$, observe that $\overline{D} = \{\{\overline{d_1\sigma}\}, \ldots, \{\overline{d_n\sigma}\}\}$. **Case 1**: If C was not used then $\overline{D}$ must contain two complementary unit clauses, so that D must contain two complementary literals, and hence $D \equiv \top$. **Case 2**: If C was used then it must occur exactly once as the top leaf in a linear resolution derivation in which literals are successively resolved away from the descendents of C using the complementary clause in $\overline{D}$. And if θ denotes the composition of the unifiers used in the derivation, then it follows that for every literal in the clause $C\theta$ (which is ground) there is a complementary unit clause in $\overline{D}$. And therefore $C\theta \subseteq D\sigma$. And since $C\theta$ is ground, and σ is a Skolemising substitution, it follows (c.f. Lemma A.7.2.3) that $C\phi\sigma \subseteq D\sigma$ where $\phi = \{X/t' \mid X/t \in \theta \text{ and } t' = t\sigma^{-1}\}$. Therefore $C\phi \subseteq D$ (c.f. Lemma A.7.2.2), and finally $C \succeq D$ by definition of θ-subsumption. ∎

[1]Lemma 2.3.1 can be seen as expressing the "subsumption algorithm" presented in [CL73] p.95 in terms of c-refutations. In addition, Lemma 2.3.1 can be seen as a special case of [Yam99] Thm4.1, in which the background theory is empty.

Proposition 2.3.2 (Equivalence of Relative Subsumption and C-Refutation - [Yam99][2]). *Given a clausal theory T and two clauses C and D, then*

$$C \succeq_T D \quad \textit{iff} \quad (T \cup \overline{D}, C) \vdash_c \Box$$

Proof. Using condition iii) of Definition 4, it is sufficient to show that

$$[D \equiv \top] \text{ or } [(T,C) \vdash_c D' \text{ and } D' \succeq D] \ \text{ iff } \ (T \cup \overline{D}, C) \vdash_c \Box$$

[**Only-If-Part**]. **Case 1**: If $D \equiv \top$ then $\overline{D}$ must contain some ground atom A and its complement $\neg A$. Hence $\Box$ is trivially derivable from $\overline{D}$ without using C or any clause in T at all. Therefore $(T \cup \overline{D}, C) \vdash_c \Box$ by Definition 3. **Case 2**: $(T,C) \vdash_c D'$ and $D' \succeq D$. From $(T,C) \vdash_c D'$ it follows by Definition 3 that there is a resolution derivation $\mathcal{R}_1$ of D' from $T \cup \{C\}$ that uses C at most once. And from $D' \succeq D$ it follows by Lemma 2.3.1 that $(\overline{D}, D') \vdash_c \Box$, and so by Definition 3 that there is a resolution derivation $\mathcal{R}_2$ of $\Box$ from $\overline{D} \cup D'$ that uses D' at most once. Combining $\mathcal{R}_1$ and $\mathcal{R}_2$ gives a resolution derivation $\mathcal{R}$ of $\Box$ from $T \cup \{C\} \cup \overline{D}$ that uses C at most once. Hence $(T \cup \overline{D}, C) \vdash_c \Box$ by Definition 3.

[**If-Part**]. If $(T \cup \overline{D}, C) \vdash_c \Box$ by Definition 3 there exists a resolution derivation $\mathcal{R}$ of $\Box$ from $T \cup \{C\} \cup \overline{D}$ that uses C at most once. **Case 1**: If C was not used in $\mathcal{R}$ then it follows from the soundness of resolution that $T \cup \overline{D} \models \Box$. Hence $T \wedge \neg D \models \bot$ by Definition 1 and $T \models D$ by the entailment theorem. Therefore, it follows from the subsumption theorem that either D is a tautology, or else there exists a resolution derivation $\mathcal{R}'$ of a clause D' from the clauses in T, such that $D' \succeq D$. And since C is not used at all in $\mathcal{R}'$ it follows from Definition 3 that $(T,C) \vdash_c D'$. **Case 2**: If C was used in $\mathcal{R}$ then it was used exactly once. Writing D explicitly as the set of literals $\{d_1, \ldots, d_n\}$, observe that $\overline{D}$ is the set of ground unit clauses $\{\{\overline{d_1\sigma}\}, \ldots, \{\overline{d_n\sigma}\}\}$. Now let $\mathcal{R}'$ be a derivation obtained from $\mathcal{R}$ by repeatedly performing the following operation. First, select any leaf clause L that is a variant of a clause $\{\overline{d_i\sigma}\}$ in $\overline{D}$. Call the child of this leaf clause Q and call the other parent of this child P (so that P and L are the parents of Q). Then, add the literal $d_i\sigma$ to each successive descendent of Q that does not already contain $d_i\sigma$ - stopping as soon as a descendent is reached that already contains $d_i\sigma$, or upon reaching the root. Finally, replace the sub-tree rooted in Q by the sub-tree rooted in P (thus eliminating the resolution step involving L). Repeat until no leaf occurrences of $\overline{D}$ remain. Note that for each L treated in this way, either the root remains unchanged (if some descendent of Q already contained $d_i\sigma$), or else the literal $d_i\sigma$ is inserted into the root (if Q was not itself the root) or else a literal d_i' subsuming $d_i\sigma$ (with some substitution θ) is inserted into the root (if Q was itself the root). Note also that no other literals are added to the root, and all other leaves are unchanged. The result is therefore a resolution derivation from the clauses $T \cup \{C\}$ and using C at most once, of a clause D' sucht that $D'\theta \subseteq D$. Hence $(T,C) \vdash_c D'$ by Definition 3 and $D' \succeq D$ by definition of θ-subsumption. ∎

[2]Proposition 2.3.2 generalisation [Yam99] Thm 4.1 by disposing of the unnecessary assumption $T \not\models D$. Note that if $T \models D$ then on the one hand: i) $T \models \forall(C\phi \rightarrow D)$ is trivially satisfied for any clause C and any substitution ϕ, and ii) $T \models \forall(D \leftrightarrow F)$ and $C \succeq F$ for any clause C by taking F as the null-clause. Thus C subsumes D relative to T. And on the other hand: if $T \models D$ then T and $\overline{D}$ are inconsistent, so that the empty-clause is derivable from them without even using C. Hence there is a c-refutation $(T \cup \overline{D}, C) \vdash_c \Box$. Consequently, Yamamoto's result can be generalised, as shown above, to include the case $T \models D$.

Knowledge Base
$K_B = \left\{ \begin{array}{l} meal(X) \text{:-} fries(X),\ burger(X) \\ burger(Y) \text{:-} fries(Y),\ offer(Y) \end{array} \right\} \cup \left\{ \begin{array}{l} bistro(md) \\ bistro(bk) \\ bistro(rz) \end{array} \right\} \cup \left\{ \begin{array}{l} offer(md) \\ offer(bk) \\ burger(rz) \end{array} \right\}$

key: md=mcDonalds, bk=burgerKing, rz=theRitz

Figure 2.1: Fast Food Example - the Knowledge Base K_B

2.4 Running Example

A running example will be used in subsequent chapters as a simple illustration of various concepts and algorithms. The knowledge base for this example, which will henceforth be denoted K_B, is shown in Figure 2.1 below. This example concerns a highly simplified domain of Fast-Food outlets, called *bistros* for short. For convenience the knowledge base K_B has been partitioned into three components.

The first component contains two rules representing **domain knowledge**. These rules state respectively that that: i) to have a *meal* in a particular *bistro* it is sufficient to have a *burger* and portion of *fries*; and ii) if the *bistro* is participating in a certain special *offer* then a free *burger* comes with every portion of *fries*.

The second component contains three facts representing **type information**, and states that there are three *bistros* called *mcDonalds*, *burgerKing* and *theRitz*, which for brevity, here and throughout, are abbreviated to *md*, *bk* and *rz*.

The third component contains three facts representing **scenario knowledge**, stating that *md* and *bk* are participating in the special *offer* described above, and a *burger* has been had at *rz*.

Chapter 3

Inductive Logic Programming and Progol

The advancement of the theory and practice of ILP is the ultimate goal of this research. In this chapter the ILP task is formalised, and the prominent Progol approach is analysed. To begin, Section 3.1 motivates and defines the ILP task. Section 3.2 then explains from first principles the theoretical concepts that underlie the Progol approach. Following that, Section 3.3 describes the influential Progol4 system and examines the recent Progol5 extension. This analysis uncovers a new source of incompleteness of the Progol5 proof procedure with respect to its intended semantics. This incompleteness will subsequently motivate a review of Abductive Logic Programming in Chapter 4, and will inspire the Hybrid Abductive Inductive Learning approach proposed in Chapter 5.

The reader is advised that the algorithms in this chapter do not follow previous accounts word for word. This is because a number of errors and omissions have been corrected, and where appropriate the algorithms have been elaborated or restructured for ease of understanding and presentation. Such points of departure, however, are clearly indicated. Instead, it is the purpose of this chapter to provide a rational reconstruction and critical analysis of the Progol approach. In particular, this chapter takes issue with the standard formulation of the ILP task, studies the relationship between Bottom Generalisation and relative subsumption, and establishes the soundness and incompleteness of the Progol5 STARTSET routine.

3.1 The Task of Inductive Logic Programming

In the field of machine learning, induction refers to the generalisation of examples with respect to prior background knowledge. It is customary to consider both positive and negative examples (i.e. examples and counter-examples). Informally, an inductive generalisation is a hypothesis that when added to the background knowledge explains or 'covers' the positive examples and is 'consistent' with the negative examples. ILP is concerned with the formalisation and automation of inductive generalisation within a logic programming context. Thus, within ILP, knowledge, examples and hypotheses are represented as formulae of first order logic, and often, for reasons of computational efficiency, clausal form logic or some more convenient subset thereof. In practice, automated approaches to ILP rely heavily upon user supplied language and search bias. In this context, language bias, which refers to syntactic constraints imposed on hypothesised clauses, is typically expressed as a hypothesis space –

i.e. a set of clauses from which hypotheses may be constructed. And search bias, which refers to procedural constraints imposed on the mechanisms used to construct those hypotheses, is usually represented by a preference criterion – i.e. an ordering used to discriminate between competing hypotheses.

In summary, the inputs to the ILP task consist of background knowledge, positive and negative examples, hypothesis space, and preference criterion. The output is a preferred hypothesis 'compatible' with the hypothesis space, that 'covers' the positive examples and is 'consistent' with the negative examples. It is usual to separate the (logical) notion of inductive generalisation from the (extra-logical) notion of preference, and to view ILP as the task of finding preferred inductive generalisations. Consequently, in order to formalise the ILP task, two things are necessary: first one must define what it means to be an inductive generalisation, and then one must define what it means for such a generalisation to be preferred. The remainder of this section does just this. First, the notion of an inductive generalisation is defined in Section 3.1.1. This definition is then compared in Section 3.1.2 with a number of variations commonly encountered in the literature, and the reasons for rejecting these alternatives are given. Sections 3.1.3 and 3.1.4 then consider in more detail the notions of language and search bias, formalising the concepts of mode-declarations and compression, as used by Progol. Finally, Section 3.1.5 examines the widely used cover-set strategy and formalises the specific ILP problem addressed in this research.

3.1.1 Inductive Generalisation

The notion of *inductive generalisation* defines logically what it means for an inductive hypothesis to generalise positive and negative examples with respect to prior background knowledge. This notion is logical in the sense that the inputs are all logical theories, and the output is a logical theory satisfying certain syntactic and semantic constraints with respect to the inputs. Here, a syntactic constraint is a restriction, possibly dependent upon the inputs, constraining those formulae that may appear in the output; and a semantic constraint is a restriction involving only the inputs, the output, and the relation of logical (i.e. semantic) entailment.

Before one can talk of an inductive generalisation, four inputs must be specified: background knowledge, positive and negative examples, and hypothesis space. It is convenient to group these inputs into a single mathematical entity, called an *inductive context* in Definition 5 below. The intention is that the background knowledge is represented by the clausal theory B, each positive example is represented by a single clause in E^+, each negative example is represented by a single clause in E^-, and the hypothesis space is represented by a set of clauses $\mathcal{H}$ – each subset of which denotes a possible hypotheses.

Definition 5 (Inductive Context). An *inductive context* is a 4-tuple $\langle B, E^+, E^-, \mathcal{H} \rangle$ where B, E^+, E^- and $\mathcal{H}$ are clausal theories. ⧫

In general, an inductive context allows arbitrary clausal theories to appear in place of B, E^+, E^- and $\mathcal{H}$. It is often necessary, however, to restrict each of these components to clausal theories of a certain type. This happens, for example, in connection with many early ILP systems, which typically require B to consist of definite clauses and E^+ and E^- to consist of ground atomic clauses. More recent ILP systems tend to be much less restrictive and often allow B, E^+, E^- and $\mathcal{H}$ to contain Horn clauses, as in the case of [Mug95], and sometimes arbitrary clausal theories, as in the case of [Ino01b, YF00].

The syntactic restrictions imposed upon an inductive context are represented by a so-called *language structure*. As formalised in Definition 6 below, a language structure consists of four languages: a language $\mathcal{L}_B$ for background knowledge, a language $\mathcal{L}_{E+}$ for positive examples, a language $\mathcal{L}_{E-}$ for negative examples, and a language $\mathcal{L}_H$ for the hypothesis space. Here, a *language* is simply a set of clausal theories: so, for example, the language of Horn clauses, denoted $\mathcal{L}_{Horn}$, is the set of all Horn clause theories. As shown below, an inductive context is compatible with a language structure if and only if each individual component lies within the corresponding language.

Definition 6 (Language Structure - [Yam99][1]). A *language structure* is a 4-tuple $\langle \mathcal{L}_B, \mathcal{L}_{E+}, \mathcal{L}_{E-}, \mathcal{L}_H \rangle$ where $\mathcal{L}_B$,$\mathcal{L}_{E+}$,$\mathcal{L}_{E-}$ and $\mathcal{L}_H$ are sets of clausal theories. An inductive context $\langle B, E^+, E^-, \mathcal{H} \rangle$ is *compatible* with a language structure $\langle \mathcal{L}_B, \mathcal{L}_{E+}, \mathcal{L}_{E-}, \mathcal{L}_H \rangle$ iff $\langle B, E^+, E^-, \mathcal{H} \rangle \in \mathcal{L}_B \times \mathcal{L}_{E+} \times \mathcal{L}_{E-} \times \mathcal{L}_H$ ♦

The concepts of inductive context and language structure relate only to the inputs of an inductive generalisation. The notion of inductive generalisation (with respect to a given inductive context) is now itself formalised in Definition 7 below. Note that this definition makes precise those notions previously introduced only informally. The 'coverage' of positive examples is defined in terms of logical entailment, 'consistency' with the negative examples is specified in terms of logical consistency, and 'compatibility' with the hypothesis space is represented by means of the subset relation. A simple illustration of this definition is given in Example 3.1.1 below.

Definition 7 (Inductive Generalisation - [MR94][2]). Given a context $\langle B, E^+, E^-, \mathcal{H} \rangle$, then an *inductive generalisation* $H \subseteq \mathcal{H}$ is a clausal theory such that the following two conditions hold:

i. $B \cup H \models E^+$ **- coverage**

ii. $B \cup H \cup E^- \not\models \Box$ **- consistency** ♦

Example 3.1.1. First, define the inductive context as follows. Let B be the knowledge base K_B shown in Figure 2.1. Let E^+ consist of the two positive unit clauses $meal(md)$ and $meal(bk)$. Let E^- consist of the single negative unit clause $\text{:-}\, meal(rz)$. Let $\mathcal{H}$ contain the clauses $fries(Z)$ and $fries(Z) \text{:-}\, offer(Z)$. Now, verify that the hypothesis H consisting of the single clause $fries(Z) \text{:-}\, offer(Z)$ is a valid inductive generalisation. As required, H is a subset of $\mathcal{H}$. The coverage condition is satisfied as B and H together entail both $meal(md)$ and $meal(bk)$. The consistency condition is satisfied as B and H are consistent with $\text{:-}\, meal(rz)$ – or equivalently – because B and H do not entail $meal(rz)$.

[1]Definition 6 is based on [Yam99] §3.1.

[2] Definition 7 is based on [MR94] Defn 3.1, where the authors specify two posterior and two prior conditions on the ILP task. The posterior conditions, which are identical to the coverage and completeness criteria above, specify the correctness of an inductive solution. The prior conditions identify two conditions under which no inductive solution can possibly exist or where none is required. But although they are semantically inconsequential, these prior conditions will later reappear in an algorithmic context: see for instance Example 3.1.7 and Footnote 5.

3.1.2 On Alternative Formulations

Before moving on to consider the notion of preferred inductive generalisations and hence of ILP itself, it is necessary to remark that several variations of Definition 7 are commonly encountered in the literature. In the following paragraphs, the most common of these alternatives are compared to Definition 7 above, and the reasons for rejecting these alternatives are given.

Hypothesis Space. Compared to Definition 7 above, a more expressive notion of language bias is sometimes encountered in which the hypothesis space is viewed not as a set of clauses, but as a set of clausal theories. Instead of being chosen from among the subsets of $\mathcal{H}$, hypotheses H are chosen from among the members of $\mathcal{H}$. This alternative formulation allows restrictions not only on which clauses may appear in H, but also on which combinations of clauses may appear in H. In practice this view is extremely uncommon due to the increased difficulty of specifying and enforcing this more expressive account of language bias, and therefore it will be pursued no further in this report.

Coverage. The condition called 'coverage' in Definition 7 above, is often referred to elsewhere as a 'completeness' condition. But while the terms 'coverage' and 'completeness' are both widely used in ILP, the latter has in addition a distinct and well defined logical meaning that will also play a prominent role this report. Therefore, in order to avoid unnecessarily overloading terms, 'coverage' is used Definition 7, and 'completeness' is reserved for its logical meaning. Note, however, that while the term 'consistency' also has a well defined logical sense, it is precisely in this sense that it is used in Definition 7 above.

Consistency. Two variations of the 'consistency' condition pervade the literature: ii.a) $B \cup H \not\models E^-$ and ii.b) $B \cup H \not\models e$ for all $e \in E^-$. It is worth noting that although ii.a is satisfied whenever ii.b is, as shown below, the converse is not generally true (so they are not equivalent). Now, taking these alternatives one at a time, first observe that condition ii.a differs logically from condition ii. of Definition 7 only in the fact that E^- is not negated. This subtle difference has important consequences, however, with respect to the reading of the negative examples. Whereas Definition 7 requires the consistency of E^- with B and H, condition ii.a requires instead that E^- not be entailed by B and H. In practice, this means that the negative examples used in the former view, correspond to the negations of the negative examples used in the latter. Thus in order to correctly represent Example 3.1.1 within this new setting, the earlier negative example $:\!-\,meal(rz)$, must be replaced by the new negative example $meal(rz)$. Although the transformation itself is unproblematic, in general condition ii.a leads to a significant difficulty, which is illustrated below.

Assume that a particular person called *arney* is known to neither *drink* nor *drive*. In the context of condition ii.a this would be represented by assigning to E^- the two clauses $drink(arney)$ and $drive(arney)$. Assume for the sake of simplicity that the background knowledge B and positive examples E^+ are both empty. Now consider the hypothesis H consisting of the single clause $drink(arney)$. The point of this example is that although intuitively this hypothesis is in blatant violation of the first negative example, it is nevertheless a valid hypothesis according to condition ii.a. This is because if B, H and E^- are interpreted as a clausal theories then condition ii.a amounts to $drink(arney) \not\models drink(arney) \wedge drive(arney)$, which is certainly true. In general, the charge is that condition ii.a fails to rule out incorrect hypotheses that violate some but not all of negative examples.

In fact, condition ii.a is very often used as if it were syntactic sugar for the stronger but more cumbersome condition ii.b. But even then, matters are only partially improved. While this latter condition correctly rules out the false hypothesis described above, it is still susceptible to another problem; which is now illustrated. Assume once again that B and E^+ are empty, and that E^- consists of the two clauses $drink(arney)$ and $drive(arney)$. But this time consider the hypothesis H consisting of the single clause $drink(arney) \vee drive(arney)$. As before, the hypothesis intuitively violates the negative examples, and yet it conforms to condition ii.b. This is because $drink(arney) \vee drive(arney) \not\models drink(arney)$ and $drink(arney) \vee drive(arney) \not\models drive(arney)$, as required. This time, the charge is that condition ii.b fails to rule out incorrect hypotheses that violate all of the negative examples collectively.

Notice that what is being argued here, is that like condition ii.a, condition ii.b fails to formalise the intuitive notion of consistency that people have in relation to inductive reasoning. Intuitively, positive and negative examples represent facts known respectively to be true and false in some particular setting – usually as a result of direct observation. The inductive task is to find a hypothesis that best accounts for these observations relative to any prior background knowledge concerning that setting. It is implicit therefore that the hypothesis has at least one model – i.e. the observed scenario – in which all of the positive examples are true and none of the negative examples are true. Clearly this possibility is precluded in the example just given, in which every model violates at least one negative example. It is now argued, however, that condition ii. of Definition 7 does give the correct behaviour.

Observe that when B and E^+ are empty and E^- contains the clauses $:\!-\, drink(arney)$ and $:\!-\, drive(arney)$, then the two false hypotheses $drink(arney)$ and $drink(arney) \vee drive(arney)$ are correctly ruled out by condition ii. In the first case $drink(arney) \wedge (\neg drink(arney) \wedge \neg drive(arney)) \not\models \bot$ is false; and in the second $(drink(arney) \vee drive(arny)) \wedge (\neg drink(arney) \wedge \neg drive(arney)) \not\models \bot$ is false. What is more, condition ii. gives the correct behaviour for general integrity constraints expressed in the form of denials. For example, if E^- contains the example $:\!-\, female(arny)$ stating that $arney$ is not $female$, and the constraint $:\!-\, male(X), female(X)$ stating that nobody is both $male$ and $female$, then, as intended, condition ii. requires the consistency of B, H and $\neg female(arney) \wedge \forall X(\neg male(X) \vee \neg female(X))$, or equivalently that $B \cup H \not\models female(arny) \vee \exists X(male(X) \wedge female(X))$. Finally, note that if E^- is empty then, as intended, condition ii. requires the consistency of B and H.

In conclusion, it is believed the above discussion lends support to the claim that the formulation of consistency adopted in Definition 7 has important advantages over those more common variations that pervade the literature. Not only does Definition 7 represent the simplest formalisation of the intended behaviour[3], in which B, H, E^+ and E^- are all interpreted as clausal theories according to the standard conventions, but it is also the case that both negative examples and integrity constraints are handled uniformly within the formalisation; the former being as a special case of the latter. It is at least hoped that this discussion has called into question some established but outdated preconceptions concerning the very formulation of the ILP task, and which have been a cause of concern, if not confusion, for more than one student of the subject.

[3]A somewhat more extreme position that arguably presents a more faithful semantics will not be propounded here. It is sufficient that Definition 7 is suitable for the purposes of this report.

Noiseless Data Assumption. Many ILP systems do not strictly enforce the coverage or consistency conditions, but instead they attempt to *maximise* the number covered of positive examples and to *minimise* the number of violated negative examples. The motivation behind this approach is that it provides a convenient mechanism for the handling of noisy data. The purpose of Definition 7, however, is to provide a *logical* account of inductive generalisation, and not, at this stage, to burden this account with additional considerations of a practical nature. Moreover, this same principle will apply generally throughout the report, whose main purpose is to provide a theoretical treatment of a novel approach to machine learning. While the techniques developed in this report can be adapted with mechanisms more suited to noisy data, this report must understandably focus on those techniques themselves.

It might also be remarked that a cautionary note is evident in the above discussion concerning the notion of consistency. An example is described above of a hypothesis that is inconsistent with all of the negative examples collectively, but consistent with each of the negative example individually. How many negative examples are violated in this case? What does it mean, therefore, to minimise the number of violated negative examples in cases like these? In order to avoid such problems, and to concentrate on the task of motivating, formalising and analysing the HAIL approach, it will be assumed throughout this report that all inputs are free of noise; and, as formalised in Definition 7, a valid hypothesis is required to cover all positive examples and to be consistent with all negative examples.

3.1.3 Language Bias and Mode-Declarations

Having decided upon the definition of inductive generalisation, the discussion returns now to the notion of language bias. As stated earlier, language bias refers to constraints on the syntactic form of hypothesised clauses, and is represented above as a hypothesis space. But, in practice, the hypothesis space is just a convenient idealisation, for it is usually difficult, or impossible, to explicitly enumerate the clauses contained therein.

Instead, a number of much more convenient representations of language bias have been developed and are currently in use. The mechanism described below, which is used both by Progol and also in many other areas of logic programming, is that of *mode-declarations* or more simply *mode-decs*. The relevant notation and terminology is now formalised in Definition 8 and illustrated in Examples 3.1.2 and 3.1.3 below.

Definition 8 (Mode-Declaration - [Mug95][4]). A *placemarker* k is either: an *in-type* $+t$, an *out-type* $-t$, or a *con-type* $\sharp t$, where t is *type predicate.* A *scheme* s is any ground atom $p(\dots)$ with the difference that placemarkers may optionally appear instead of constant symbols. A *mode-declaration* or *mode-dec* m is either a *head-dec* $\mathsf{modeh}[r,s]$ or a *body-dec* $\mathsf{modeb}[r,s]$, where $r \geq 1$ is a natural number called the *recall* and s is a scheme.

If M is a set of mode-decs, then M^+ and M^- denote the set of head and body-decs in M. If m is a mode-dec, $recall(m)$ is the recall of m, $pred(m)$ is predicate symbol in the scheme of m, $schema(m)$ is the scheme of m with every placemarker occurance k_i replaced by a *fresh* variable X_i, and $type(m)$ is the set of corresponding atoms $t_i(X_i)$, where t_i is the type predicate in k_i and X_i is the variable that replaced k_i. If k_i is an in-type placemarker then X_i is called an in-type variable or simply a +type, and similarly for out- and con-types

[4]Definition 8 is based on [Mug95] Def20. The terminiolgy *schema* is used in place of *atom* and the terminology 'compatible with' is used instead of 'in the language of'.

Background	Positive Examples	Negative Examples
$B = K_B$	$E^+ = \left\{ \begin{array}{c} meal(md) \\ meal(bk) \end{array} \right\}$	$E^- = \{ \text{:-}\, meal(rz) \}$

Mode Declarations
$M = \left\{ \begin{array}{l} \mathsf{modeh}[fries(+bistro)] \\ \mathsf{modeb}[offer(+bistro)] \end{array} \right\}$

$\Rrightarrow$

Hypothesis
$H = \{fries(Z) \text{:-}\, offer(Z)\}$

Figure 3.1: Fast-Food Example: an ILP Problem

A clause C is *compatible* with M iff there is a mapping from the head/body atoms in C to the head/body-decs in M such that each atom is subsumed by the schema of corresponding mode-dec and the following conditions are satisfied: i) every $-$type is bound to a variable, ii) every $+$type is bound to a variable that in some head atom is bound by an $+$type, or in some *preceding* body atom is bound by a $-$type, and iii) every $\sharp$type is bound to a constant of the correct type. Finally, the hypothesis space $\mathcal{H}_M$ *determined* by M is the set of non-empty reduced clauses compatible with M. ♦

Example 3.1.2. Let M consist of the head-decs $\mathsf{modeh}[fries(+bistro)]$ and the body-dec $\mathsf{modeb}[offer(+bistro)]$. Then the hypothesis space $\mathcal{H}_M$ consists of the clauses $fries(Z)$ and $fries(Z) \text{:-}\, offer(Z)$ where Z is a variable.

Example 3.1.3. Let M' consist of the head-dec $\mathsf{modeh}[p(a, f(\sharp any), +any)]$ and the body-dec $\mathsf{modeb}[q(+any, g(-any))]$. If the constants a and b have type any then clause $H' = p(a, f(b), X) \text{:-}\, q(X, g(Y)), q(Y, g(X))$ is compatible with M'. But if the order of the two body atoms were to be reversed then the resulting clause would no longer be compatible.

In view of their convenience as a representation of language bias, it is desirable to incorporate the notion of mode-declaration directly within the formalism of an inductive context. A *mode context* is defined, therefore, as an inductive context except that the hypothesis space $\mathcal{H}$ is not enumerated explicitly, but is represented by a set of mode-declarations M. In this case, the hypothesis space $\mathcal{H}$ is understood as being the hypothesis space $\mathcal{H}_M$ determined by M. Finally, mode contexts in which consist entirely of Horn clauses will be called *Horn contexts*, and will be used frequently in the sequel. To illustrate these concepts, Figure 3.1 represents the same inductive task as used previously in Example 3.1.1. Note that for convenience here and throughout this report, recalls are omitted and are assumed to be arbitrarily large.

3.1.4 Search Bias and Compression

As described previously, search bias refers to procedural constraints on the mechanisms used to construct inductive generalisations, and is often represented as a preferential ordering among possible hypotheses. In practice, preferential selection is usually achieved through the use of preference heuristics during the construction of hypotheses. The heuristic that used by Progol is popularly called *compression* and takes into account the number of covered positive examples, the number of negative examples violated, and a syntactic measure of the complexity of the hypothesis.

In connection with Definition 7 above, it is appropriate to assign a compression of $-\infty$ to any hypothesis H that together with background knowledge B is inconsistent with the positive and negative examples E^+ and E^-. For if this were not the case, then H could not be a valid inductive generalisation – violating one or both of the required conditions. In all other cases, compression is defined as the 'coverage' of E^+ less the 'complexity' of H, where coverage is the number of positive examples entailed by B and H, and complexity is the number of literals in H. These notions are formalised in Definitions 9, 10 11 and 12 below.

Definition 9 (Compression).

$$Compression(H, B, E^+, E^-) =$$

$$\begin{cases} -\infty & \text{if } B \cup H \cup E^+ \cup E^- \models \Box \\ Coverage(B \cup H, E^+) - Complexity(H) & \text{if } B \cup H \cup E^+ \cup E^- \not\models \Box \end{cases}$$

♦

Definition 10 (Coverage).

$$Coverage(B, E) = |Cover(B, E)|$$

♦

Definition 11 (Cover).

$$Cover(B, E) = \{e \in E \mid B \models e\}$$

♦

Definition 12 (Complexity).

$$Complexity(H) = |H_1| + \cdots + |H_n| \text{ where } H = \{H_1, \ldots, H_n\}$$

♦

Formally, the compression metric is closely related to principle called *Minimum Description Length*, which in turn is related to the notions of maximum-likelihood, coding length and entropy. In this report it will be sufficient to observe that compression prefers consistent hypotheses with few literals covering many positive examples. By Definitions 7 and 9, the compression of a valid inductive generalisation must equal the number of positive examples less the number of literals in the hypothesis. Consequently, preferred hypotheses are those with the fewest number of literals. These concepts are now illustrated in Example 3.1.4 below.

Example 3.1.4. Referring to Figure 3.1 above, it is easily seen that the hypothesis $H = \{fries(Z) \text{:-} offer(Z)\}$ has a coverage of 2, a complexity of 2, and therefore a compression 0. This hypothesis is favoured by the compression metric as no other valid hypothesis has fewer literals. Note that the hypothesis $H' = \{fries(Z)\}$ is *not* valid because it violates the consistency requirement of Definition 7, and it has a compression of $-\infty$.

3.1.5 The Cover-Set Principle

The necessary machinery is now in place so that ILP may be defined as the task of finding maximally preferred inductive generalisations. The only reason why a slightly less direct approach must instead be taken, is that in practice it is a highly intractable task to search for such global maxima. The recourse most often adopted in practice is that of greedily constructing hypotheses in a piecemeal fashion using a technique known as the *cover-set* principle (see for example [Mic84]). The motivation is that it is generally easier to construct several smaller *sub-hypotheses* each covering a few examples at a time, than it is to construct one larger hypothesis covering all of the examples in one go.

The idea is to construct a hypothesis H by adding together a succession of smaller sub-hypotheses H_i each of which extends the coverage of E^+ and maintains consistency with E^-. A so-called *cover-set-loop* constructs one sub-hypothesis H_i on every iteration, until all positive examples have been covered. Since each H_i must cover at least one as yet uncovered positive example, it is common at the beginning of each iteration to arbitrarily select one example e_i, called a *seed example*, and to search for the most compressive H_i covering this example at the very least. After its construction H_i is usually asserted into the background knowledge B and any newly covered positive examples (including e_i) are removed from E^+, ready for the next iteration.

These notions are now formalised in Definitions 13, 14 and 15 below. A *selection function* formalises the process of selecting a seed example from a set of positive examples. An *inference method* formalises the method used to derive the sub-hypotheses H_i. Finally, the ILP task is formulated according to the cover-set approach described above. Conditions i. and ii. require that each H_i entails the seed example e_i relative to the current background B_i, and is consistent with *all* negative and positive examples E^+ and E^-. Condition iii. requires that no other such hypothesis which can be derived by $\mathcal{I}$ has greater compression. Lastly, condition iv. requires that upon termination no positive examples remain.

Definition 13 (Selection Function). Given a set of clauses E^+, a *selection function* π maps each non-empty subset of E^+ to one member, called a *seed* example, of that subset. ♦

Definition 14 (Inference Method). Given a clausal theory B and a clause e, an *inference method* $\mathcal{I}$ defines a set of clausal theories, denoted $\mathcal{I}(B, e)$ and called the *hypotheses derivable by $\mathcal{I}$ from B and e*, where each such hypothesis H entails e relative to B (i.e. $B \cup H \models e$). Given a set M of mode-decs, the notation $\mathcal{I}_M(B, e)$ will denote the set of hypotheses derivable by $\mathcal{I}$ from B and e that are compatible with M (i.e. $\mathcal{I}_M(B, e) = \{H \in \mathcal{I}(B, e) \mid H \subseteq \mathcal{H}_M\}$). ♦

Definition 15 (ILP). Given as input an inductive context $\langle B, E^+, E^-, M\rangle$, a selection function π, and an inference method $\mathcal{I}$, the task of ILP is to return as output a clausal theory $B \cup H$ where $H = H_1 \cup \cdots \cup H_n$ is such that for all $1 \leq i \leq n$:

i. $H_i \in \mathcal{I}_M(B_i, e_i)$

ii. $compression(H_i, B_i, E_i^+, E^-) > -\infty$

iii. $compression(H_i, B_i, E_i^+, E^-) \geq compression(H_i', B_i, E_i^+, E^-)$ for all $H_i' \in \mathcal{I}_M(B_i, e_i)$

iv. $E_{n+1}^+ = \emptyset$

where, by definition:

a. $B_i = B \cup H_1 \cup \cdots \cup H_{i-1}$ for $1 \leq i \leq n+1$ (n.b. B_1 is identical to B)

b. $E_i^+ = E^+ - Cover(B_i, E^+)$ for $1 \leq i \leq n+1$ (n.b. E_1^+ are those E^+ not covered by B)

c. $e_i = \pi(E_i^+)$ for $1 \leq i \leq n$ (n.b. for e_i to be defined, these E_i^+ are non-empty) ♦

Informally, condition i. requires that the sub-hypothesis H_i output on the ith iteration is derivable by the chosen inference method $\mathcal{I}$, and therefore covers at least the seed example e_i relative to the current background B_i. Condition ii. essentially requires that H_i be consistent with the background knowledge and the examples (failing which is the only way the compression could be $-\infty$). Condition iii. requires that no more compressive such hypothesis can be derived. Finally, condition iv. requires that eventually all positive examples are covered.

A simple illustration of the definitions above is provided in Example 3.1.5 below. The hypothesis H consists of two sub-hypotheses H_1 and H_2, which in practice would be produced on two successive iterations of a cover-set-loop. It is easily verified that H satisfies the requirements i-iv. above. Initially, background $B_1 = B$, positive examples $E_1^+ = E^+$, and seed example $e_i = p$ is selected by π. The hypothesis $H_i = \{a\}$ is the most compressive clause covering e_1 that can be derived by Bottom Generalisation. The clause a is therefore inserted into B_2 and the covered example p is removed from E_2. Next, $e_2 = q$ is chosen and $H_2 = \{b\}$ is constructed. The clause b is therefore added to B_3, both covered examples are removed from E_3^+, which is now empty.

Example 3.1.5. Let π select examples in ascending alphabetic order, let $\mathcal{I}$ be the inference method of Bottom Generalisation, and let the background, examples and mode-decs be defined as follows:

$$B = \left\{ \begin{array}{l} p \text{:-} a \\ q \text{:-} b \\ r \text{:-} a, b \end{array} \right\} \qquad E^+ = \left\{ \begin{array}{c} p \\ q \\ r \end{array} \right\} \qquad E^- = \emptyset \qquad M = \left\{ \begin{array}{c} \mathsf{modeh}[a] \\ \mathsf{modeh}[b] \end{array} \right\}$$

Then, with the aid of the following diagram, verify that the hypothesis H containing the two unit clauses a and b is a valid ILP hypothesis.

iteration 1	$B_1 = \left\{ \begin{array}{l} p \text{:-} a \\ q \text{:-} b \\ r \text{:-} a, b \end{array} \right\}$	$E_1^+ = \left\{ \begin{array}{c} p \\ q \\ r \end{array} \right\}$	$e_1 = p$	$H_1 = \{a\}$
iteration 2	$B_2 = \left\{ \begin{array}{l} p \text{:-} a \\ q \text{:-} b \\ r \text{:-} a, b \end{array} \right\} \cup \{ a \}$	$E_2^+ = \left\{ \begin{array}{c} q \\ r \end{array} \right\}$	$e_2 = q$	$H_2 = \{b\}$
final	$B_3 = \left\{ \begin{array}{l} p \text{:-} a \\ q \text{:-} b \\ r \text{:-} a, b \end{array} \right\} \cup \left\{ \begin{array}{c} a \\ b \end{array} \right\}$	$E_3^+ = \emptyset$	$H = \left\{ \begin{array}{c} a \\ b \end{array} \right\}$	

The usefulness of the cover-set approach is that it enables the construction of inductive generalisations covering many positive examples, while using inference methods that take advantage of a single seed example to achieve a more focussed search. While this approach is liable to miss some more compressive hypotheses, the gain in tractability is a significant practical advantage. Proposition 3.1.6 below now proves the simple but important result that any hypothesis constructed by the ILP cover-set approach is a valid inductive generalisation.

Proposition 3.1.6. *Given an inductive context $\langle B, E^+, E^-, M \rangle$, a selection function π, and an inductive inference method $\mathcal{I}$, any theory H satisfying the ILP conditions of Definition 15, is a valid inductive generalisation by Definition 7.*

Begin Algorithm 3.1.7 (MSH-COVERSET).

```
Input              given an inductive context $\langle B, E^+, E^-, M\rangle$
                    if $B \cup E^+ \cup E^- \models \Box$ then abort
                    let $E^+ = E^+ - Cover(B, E^+)$
Cover-Set-Loop      while $E^+ \neq \emptyset$
select seed            choose seed example $e \in E^+$
construct MSH          find most *specific* hypothesis $MSH(B, e)$ such that
                           - $B \cup MSH(B, e) \models e$
generalise MSH         find most *compressive* generalisation $H_i \subseteq \mathcal{H}_M$ such that
                           - $H_i \models MSH(B, e)$
                           - $B \cup H_i \cup E^+ \cup E^- \not\models \Box$
                       if $Compression(H_i, B, E^+, E^-) = -\infty$ then *abort*
add hypothesis         let $B = B \cup H_i$
remove cover           let $E^+ = E^+ - Cover(B, E^+)$
Output             return $B$
```

End Algorithm (MSH-COVERSET)

Proof. It is sufficient to show that the three requirements of Definition 7 are satisfied whenever the five conditions of Definition 15 are. [**language**] From i. above $H_i \in \mathcal{I}_M(B_i, e_i)$ for all $1 \leq i \leq n$, and so $H_i \subseteq \mathcal{H}_M$ by Definition 14, and because $H = H_1 \cup \cdots \cup H_n$ it follows that $H \subseteq \mathcal{H}_M$. [**coverage**] From iv. above $E^+_{n+1} = \emptyset$, and so $E^+ - Cover(B_{n+1}, E^+) = \emptyset$ by Definition b. above, and therefore $B_{n+1} \models E^+$ by Definition 11, but $B_{n+1} = B \cup H$ by Definition a. above, and hence $B \cup H \models E^+$. [**consistency**] From ii. above it follows taking $i = n$ that $compression(H_i, B_i, E^+_i, E^-) > -\infty$. Therefore $B_n \cup H_n \cup E^+_n \cup E^- \not\models \Box$ by Definition 9. But $B_n \cup H_n = B \cup H$ by Definition a. above, and so $B \cup H \cup E^+_n \cup E^- \not\models \Box$, and so $B \cup H \cup E^- \not\models \Box$. ▮

The key step in the Cover-Set-Loop is clearly the construction of the sub-hypothesis H_i. In principal this step can be achieved in a variety of different ways, but in practice one of two main and approaches is usually adopted: top-down and bottom up. Intuitively, the top-down approach involves progressively specialising a very general hypothesis, while the bottom-up approach involves progressively generalising a very specific hypothesis. It turns out that the latter provides an ideal setting for the inference method of Bottom Generalisation, and therefore only the bottom-up approach will be investigated further in this report.

From this point on, the bottom-up approach will be referred to as the approach of *Most Specific Hypothesis Generalisation* in order to emphasise the two steps involved. Given background knowledge B_i and seed example e_i, the first step involves the construction of the *Most Specific Hypothesis* (MSH), denoted $MSH(B_i, e_i)$, which is a maximally specific hypothesis – according to some given generality ordering – that logically entails e_i relative to B_i. The second step involves constructing the most compressive hypothesis H_i – according to some given measure of compression – that logically entails the MSH.

A generic framework for MSH-based Cover-Set Generalisation is shown in Algorithm 3.1.7. Given an inductive context $\langle B, E^+, E^-, M\rangle$, each iteration of the cover-set-loop adds one additional sub-hypothesis H_i to B and removes one or more covered examples from E^+. But instead of returning the overall hypothesis H found by taking the union $H_1 \cup \cdots \cup H_n$ of the sub-hypotheses H_i, this routine follows standard convention by returning the augmented background theory $B \cup H$. The motivation is that having assimilated the information from the examples, this new theory will be used in future learning or reasoning problems.

The symbols B, E^+ and e are not subscripted in order to make for easier going later on. The sub-hypothesis H_i is subscripted in order to avoid confusion with the overall hypothesis H mentioned in the text above. But because the symbol appears nowhere in the actual algorithm, in future algorithms this subscript will be safely omitted for simplicity.

Prior to entering the cover-set-loop, the inputs are first checked for consistency and then any covered examples are removed from E^+. These precautions[5] avoid performing unnecessary work when no hypothesis can possibly exist, and by removing positive examples for which no hypothesis is required. Because E^+ is finite, the cover-set-loop will terminate; either with a valid inductive generalisation, or by failing to find a suitable hypothesis.

[5]These actions correspond to the prior conditions stated in [MR94] and referred to earlier in Footnote 2.

3.2 A Rational Reconstruction of the Principles of Progol

Progol is a well-known and highly successful ILP system. The purpose of this section is to present from first principles the key theoretical techniques that underlie the Progol approach, and thereby clarify the relationship between the ILP semantics formalised in the previous section and the Progol proof procedure to be described in the next. First, Section 3.2.1 recalls the inference methods of Inverse Entailment and Bottom Generalisation. This is followed in Section 3.2.2 by a discussion relating Bottom Generalisation relative subsumption. The concept of a vacuous literal is introduced in Section 3.2.3, and a standardisation technique for Horn clauses is analysed in Section 3.2.4. A refinement of Bottom Generalisation is then introduced in Section 3.2.5, and is combined with the cover-set principle in Section 3.2.6.

3.2.1 Inverse Entailment and Bottom Generalisation

The ILP approach of *Inverse Entailment* was first introduced in [Mug95] as the basis of the Progol proof procedure. The motivation behind this approach derives from the principle of IE, which in the context of ILP can be written $B \wedge H \models e$ iff $B \wedge \neg e \models \neg H$ for any formulae B, H and e. Informally, this equivalence states that the negations of inductive hypotheses may be deduced from the background knowledge together with the negation of a seed example. By exposing the close connection between induction and deduction, this equivalence suggests that deductive techniques can be brought to bear on the tasks of inductive reasoning; and one successful means of doing just this is the inference method of Inverse Entailment.

The inference method of Inverse Entailment (which this report is careful to distinguish from the Principle of IE) is based on an important theoretical construction called the *Bottom Set* or *Bottom Clause*. As formalised in Definition 16 below, given background knowledge B and a seed example e, the Bottom Set of B and e, which is denoted $Bot(B, e)$, is defined as the clause containing all those ground literals whose negations may be deduced from B together with the clausal complement of e. In addition, as formalised in Definition 17 below, a clause h is said to be *derived from B and e by Inverse Entailment* if and only if h logically entails $Bot(B, e)$ and does not contain any Skolem constants introduced therein.

Definition 16 (Bottom Set - [Mug95][6]). Given a clausal theory B and a clause e. Recall that $\mathcal{GL}$ denotes the set of ground literals. Assuming that no Skolem constant introduced in $\overline{e}$ is contained in B, the *BottomSet of B and e* is the clause denoted $Bot(B, e)$ and defined

$$Bot(B, e) = \{L \in \mathcal{GL} \mid B \cup \overline{e} \models \neg L\} \qquad \blacklozenge$$

Definition 17 (Inverse Entailment - [Mug95][7]). Given a clausal theory B and a clause e, a clause h is said to be derivable from B and e by Inverse Entailment iff h contains no Skolem constant introduced in $Bot(B, e)$ and $h \models Bot(B, e)$. $\blacklozenge$

[6]The Bottom Set concept originally appeared in [Mug95] with the name 'Most Specific Clause' and the notation '$\perp$'. The more explicit notation '$\perp(B, E)$' was used in [MB00]. Finally, the terminology 'Bottom Set' and the notation '$Bot(B, E)$' was introduced in [Yam96] and is now standard. Other variations encountered in the literature include the notation $\perp(E, B)$, $Bot(E, B)$ and the terminology 'Most Specific Hypothesis'.

[7] The inference method of 'Inverse Entailment' was introduced in [Mug95], where it was believed to constitute a complete procedure for inverting entailment between clauses relative to a clausal theory. That this belief was *mistaken* was first revealed in [Yam96] and is shown below to result from the so-called 'law of contraposition' $B \wedge H \models E$ iff $B \wedge \overline{E} \models \overline{H}$. The invalidity of this statement is easily seen by taking $B = \emptyset$, $H = p(X)$ and $E = p(a)$, where a is a constant. Taking complements $\overline{E} = \neg p(a)$ and $\overline{H} = \neg p(k)$, where k is a fresh Skolem constant. Clearly $B \wedge H \models E$ but $B \wedge \overline{E} \not\models \overline{H}$.

The key result exploited by inductive learners, is that any clause h derived from B and e by Inverse Entailment implies e relative to B, as shown in Proposition 3.2.1 below. This soundness result is computationally significant because the task of finding h that entail $Bot(B,e)$ is far more tractable than the task of finding h that entail e relative to B. First, the problem of constructing $Bot(B,e)$ involves computing only ground unit consequences of a clausal theory. Second, the problem of finding h is one of straight rather than relative inverse implication.

Proposition 3.2.1 (Soundness of Inverse Entailment - [Mug95, Yam99][8]). *Given a clausal theory B and a clause e. If h is any clause containing no Skolem constants introduced in $Bot(B,e)$, then*

$$h \models Bot(B,e) \textit{ implies } B \wedge h \models e$$

Proof. If $h \models Bot(B,e)$ then $h \models \bigvee\{L \in \mathcal{GL} \mid B \wedge \overline{e} \models \neg L\}$ by Definition 16. By the principle of IE it follows $\bigwedge\{M \in \mathcal{GL} \mid B \wedge \overline{e} \models M\} \models \neg h$ (where each M is the complement of some corresponding L). And since $B \wedge \overline{e} \models \bigwedge\{M \in \mathcal{GL} \mid B \wedge \overline{e} \models M\}$ by transitivity $B \wedge \overline{e} \models \neg h$. Since h contains no Skolem symbol in $\overline{e}$, by Definition 1 it follows $B \wedge \neg e \models \neg h$. Therefore $B \wedge h \models e$ by another application of the principle of IE. ∎

Unfortunately, what might be gained in terms of tractability is ultimately paid for by a loss of completeness (with respect to inverse relative implication). More precisely, although every hypothesis h derived by Inverse Entailment from B and e correctly entails e relative to B, not all such hypotheses can be constructed in this way, as shown in Proposition 3.2.2 below. It turns out the hypotheses which can be derived by generalising the Bottom Set are related to Plotkin's notion of relative subsumption, as will be investigated later in this section.

Proposition 3.2.2 (Incompleteness Inverse Entailment - [Yam96][9]). *There exist a clausal theory B, a clause e, and a clause h containing no Skolem constant from $Bot(B,e)$, such that $B \wedge h \models e$, but $h \not\models Bot(B,e)$.*

Proof. First, let

$$B = \left\{ \begin{array}{l} even(0) \\ even(s(X)) \text{:-}\, odd(X) \end{array} \right\} \qquad e = odd(s(s(s(0)))) \qquad h = odd(s(X)) \text{:-}\, even(X)$$

Then, verify

$$Bot(B,e) \equiv odd(s(s(s(0)))) \text{:-}\, even(0)$$

Finally, observe

$$B \wedge h \models e \text{ but } h \not\models Bot(B,e)$$ ∎

Because it is generally highly intractable to compute all h that entail a given $Bot(B,e)$, in practice the entailment relation is usually weakened to that of θ-subsumption, to result in the semantics of *Bottom Generalisation* [Mug95, Yam99], formalised in Definition 18 below. Because the hypotheses derivable by Bottom Generalisation are a strict subset of those derivable by Inverse Entailment, the soundness and incompleteness of Bottom Generalisation are immediate from Propositions 3.2.1 and 3.2.2 above.

[8]The proof of Proposition 3.2.1 was first attempted in [Mug95], but for the reason stated in Footnote 7 above, the argument therein is invalid (as are its elaborations in [MF01] and elsewhere). A correct proof of Proposition 3.2.1 is given in [Yam99], but is considerably more involved than the one given above.

[9]The counter-example in Proposition 3.2.2 is due to [Yam96] §4.

Definition 18 (Bottom Generalisation - [Mug95, Yam99][10]). Given a clausal theory B and a clause e, a clause h is *derivable by Bottom Generalisation* iff h contains no Skolem constant introduced in $Bot(B, e)$ and

$$h \succeq Bot(B, e) \qquad \blacklozenge$$

As shown in Example 3.2.3 below, the inductive problem depicted earlier in Figure 3.1 is solved by Bottom Generalisation. In this example, the first positive example has been arbitrarily chosen as the seed example, although in this case the other example would have served equally well. This simple example only hints at the potential usefulness of Bottom Generalisation in the context of ILP. Indeed, by combining this approach with the cover set principle, a far more powerful method of ILP will be developed later in this section.

Example 3.2.3. Referring to Figure 3.1, let B be the knowledge base K_B defined in Figure 2.1, and let e be the clause $meal(md)$. Observe that $Bot(B, e) = meal(md), fries(md) \text{:-} burger(rz), offer(md), offer(bk), bistro(md), bistro(bk), bistro(rz)$ and verify that the hypothesis $h = fries(Z) \text{:-} offer(Z)$ is derived by Bottom Generalisation from B and e. This is because $h \succeq Bot(B, e)$, confirming that $B \wedge h \models e$.

Notice that when θ-subsumption is used as the generalisation relation, then hypotheses may be efficiently computed by searching of the θ-subsumption sub-lattice $[\Box, Bot(B, e)]$, which is both smaller and better structured than the original search space. Notice also that if the θ-subsumption lattice is viewed as a generality ordering on clauses, with more general clauses at the bottom and more specific clauses at the top, then $Bot(B, e)$ denotes the most specific clause that entails e relative to B – hence the name "Bottom Set".

3.2.2 On Bottom Generalisation and Relative Subsumption

The present discussion concerning the soundness and completeness of Bottom Generalisation is concluded below with a technical analysis of the equivalence of Bottom Generalisation and Plotkin's relative subsumption. This important result was first shown in [Yam99], though in a form slightly less general and less intuitive than is presented below. Compared to [Yam99], additional generality is achieved below by disposing of an unnecessary assumption, and the intuition is strengthened by emphasising the clear and direct role played by the Bottom set in each of the three equivalent characterisations of relative subsumption in Definition 4. For this reason, three independent proofs are presented below.

i) First Form of Relative Subsumption: $B \models \forall(h\phi \rightarrow e)$

The formulation $B \models \forall(h\phi \rightarrow e)$ was introduced in [Plo71] as a convenient characterisation of relative subsumption. Using this formulation the task of deciding relative subsumption amounts to finding a suitable substitution ϕ satisfying this condition. What is apparent from Proposition 3.2.4 below (see only-if-part), is that if h is known to θ-subsume $Bot(B, e)$, with substitution θ, then a suitable ϕ is determined by applying to θ the inverse of the Skolemising substitution σ that was used in the construction of $Bot(B, e)$. Informally, $\phi = \theta\sigma^{-1}$.

[10]The proof procedure now called 'Bottom Generalisation' was introduced in [Mug95]. It was referred to as the inference method of 'Inverse Entailment' in [Yam97], and then as 'Bottom Generalisation' in [Yam99].

Proposition 3.2.4 (Correspondence of Bottom Generalisation and Relative Subsumption - i). *Given a clausal theory B, a clause e, and a clause h containing no Skolem constants introduced in $Bot(B,e)$, then*

$$h \succcurlyeq Bot(B,e) \quad \textit{iff} \quad B \models \forall(h\phi \rightarrow e)$$

Proof. [**Only-If-Part**]. If $h \succcurlyeq Bot(B,e)$ then by definition of θ-subsumption it follows $h\theta \subseteq Bot(B,e)$, noting that θ is a grounding substitution for h because $Bot(B,e)$ is a ground clause. Writing h explicitly as the set of literals $\{H_1,\ldots,H_n\}$, it follows that $H_i \in Bot(B,e)$ for all $1 \leq i \leq n$ and so $B \wedge \overline{e} \models \neg H_i\theta$ by Definition 16. If σ denotes the Skolemising substitution used in the complement, this can be equivalently written $B \wedge \neg e\sigma \models \neg H_i\theta$, and so $B \wedge H_i\theta \models e\sigma$ by the principle of IE. Therefore $B \wedge h\theta \models e\sigma$ (c.f. disjunction-elimination). Therefore $B \models h\theta \rightarrow e\sigma$ by the entailment theorem. Now, since $h\theta$ is ground, and σ is a Skolemising substitution, it follows (c.f. Lemma A.7.2.3) that $h\theta = h\phi\sigma$ where $\phi = \{X/t' \mid X/t \in \theta \text{ and } t' = t\sigma^{-1}\}$. And so $B \models (h\phi)\sigma \rightarrow e\sigma$ and $B \models (h\phi \rightarrow e)\sigma$ using elementary properties of substitutions. And since neither h nor e contain any Skolem constants mentioned in σ (the former by assumption, the latter by definition), it follows $B \models \forall(h\phi \rightarrow e)$ (c.f. universal-introduction). [**If-Part**]. If $h \succcurlyeq_B e$ then $B \models \forall(h\phi \rightarrow e)$ by Definition 4. If σ is a Skolemising substitution mapping each variable in $h\phi$ and e to a fresh Skolem constant then $B \models (h\phi \rightarrow e)\sigma$ (c.f. universal-instantiation). Writing θ for $\phi\sigma$ gives $B \models h\theta \rightarrow e\sigma$ where $h\theta$ and $e\sigma$ are both ground. And so $B \models \neg e\sigma \rightarrow \neg h\theta$ by taking the contrapositive. By the entailment theorem $B \wedge \neg e\sigma \models \neg h\theta$ which by Definition 1 can be written $B \cup \overline{e_\sigma} \models \neg h\theta$ using the fact that σ is a Skolemising substitution for e. Writing $h = \{H_1,\ldots,H_n\}$ where the H_i are literals, $B \cup \overline{e_\sigma} \models \bigwedge\{\neg H_1\theta,\ldots,\neg H_n\theta\}$. And since $B \cup \overline{e_\sigma}$ entails the RHS conjunction it must therefore entail each conjunct individually, and so $\{\neg H_1\theta,\ldots,\neg H_n\theta\} \subseteq \{M \in \mathcal{GL} \mid B \cup \overline{e} \models M\}$. Negating all elements gives $\{H_1\theta,\ldots,H_n\theta\} \subseteq \{L \in \mathcal{GL} \mid B \cup \overline{e} \models \overline{L}\}$ and so $h\theta \succcurlyeq Bot(B,e)$ by definition of θ-subsumption. And finally, since $h\theta$ is a ground instance of h it follows by transitivity of subsumption that $h \succcurlyeq Bot(B,e)$. ∎

ii) Second Form of Relative Subsumption: $B \models \forall(e \leftrightarrow f)$ **and** $h \succcurlyeq f$

The formulation $B \models \forall(e \leftrightarrow f)$ and $h \succcurlyeq f$ is the original characterisation of relative subsumption introduced in [Plo71]. Using this formulation the task of deciding relative subsumption amounts to finding a suitable clause f satisfying this condition. What is apparent from Proposition 3.2.7 below (see only-if-part), is that such an f is found by applying to $Bot(B,e)$ the inverse of the Skolemising substitution σ employed in $Bot(B,e)$ – giving what is called the *Generalised Bottom Set* in Definition 19 and Lemmas 3.2.5 and 3.2.6 below.

Definition 19 (Generalised Bottom-Set). Given a clausal theory B and a clause e, let $Bot(B,e)$ denote the Bottom Set of B and e, and let σ be the Skolemising substitution used in $Bot(B,e)$. The *Generalised Bottom Set of B and e*, denoted $Bot'(B,e)$, is defined as the clause uniquely obtained from $Bot(B,e)$ by replacing each Skolem constant k by the corresponding variable V for each $V/k \in \sigma$. Thus

$$Bot'(B,e) = Bot(B,e)\sigma^{-1}$$ ♦

The clear and intuitive relationship between the Generalised Bottom Set and the formula f in the above definition of relative subsumption is now clarified in Lemmas 3.2.5 and 3.2.6

below. But first of all, note that because $Bot(B,e)$ is ground and σ binds distinct variables to distinct Skolem constants, it follows $Bot'(B,e)\sigma = Bot(B,e)$. And it therefore follows that $Bot'(B,e)\sigma \succeq Bot(B,e)$ and $Bot'(B,e)\sigma \models Bot(B,e)$ by definition of θ-subsumption.

Lemma 3.2.5. *Given a clausal theory B and a clause e, let $Bot'(B,e)$ denote Generalised Bottom Set of b and e, then*

$$B \models \forall(e \leftrightarrow Bot'(B,e))$$

Proof. [$\leftarrow$**Direction**]. Recall from Definition 16 that $Bot(B,e) \equiv \bigvee\{L \in \mathcal{GL} \mid B \cup \overline{e} \models \neg L\}$ by, and so $\neg Bot(B,e) \equiv \bigwedge\{M \in \mathcal{GL} \mid B \cup \overline{e} \models M\}$ upon negation. Since $B \wedge \overline{e} \models \bigwedge\{M \in \mathcal{GL} \mid B \cup \overline{e} \models M\}$ it follows by transitivity $B \wedge \overline{e} \models \neg Bot(B,e)$. Equivalently $B \wedge \neg e\sigma \models \neg Bot(B,e)$ by definition of clausal complements. Therefore $B \models \neg e\sigma \rightarrow \neg Bot(B,e)$ by the entailment theorem, and $B \models Bot(B,e) \rightarrow e\sigma$ taking the contrapositive. And so $B \models Bot'(B,e)\sigma \rightarrow e\sigma$ from the argument above. Hence $B \models (Bot'(B,e) \rightarrow e)\sigma$. Now, since σ binds all of the variables in $Bot'(B,e)$ and e to Skolem constants not appearing in $Bot'(B,e)$ or B or e, it follows (c.f. universal-introduction) that $B \models \forall(Bot'(B,e) \rightarrow e)$. [$\rightarrow$**Direction**]. Writing $e = \{E_1, \ldots, E_n\}$ it follows $\overline{e_\sigma} \equiv \neg E_1\sigma \wedge \ldots \wedge \neg E_n\sigma$ by definition of clausal complements. Hence $\overline{e_\sigma} \models \neg E_1\sigma \wedge \ldots \wedge \neg E_n\sigma$ trivially, and $B \cup \overline{e_\sigma} \models \neg E_1\sigma \wedge \ldots \wedge \neg E_n\sigma$ by monotonicity. Now, since the LHS entails the RHS conjunction, it must entail each conjunct individually and so $\{E_1\sigma, \ldots, E_n\sigma\} \subseteq \{L \in \mathcal{GL} \mid B \cup \overline{e} \models \neg L\}$. And so by Definition 16 it follows $e\sigma \subseteq Bot(B,e)$. Hence $e\sigma \subseteq Bot'(B,e)\sigma$ by the argument given above. Therefore $e\sigma \succeq Bot'(B,e)\sigma$ and $e\sigma \models Bot'(B,e)\sigma$ by definition of θ-subsumption. Consequently $B \wedge e\sigma \models Bot'(B,e)\sigma$ by monotonicity and $B \models e\sigma \rightarrow Bot'(B,e)\sigma$ by the entailment theorem. And since σ binds all of the variables in $Bot'(B,e)$ and e to Skolem constants not appearing in $Bot'(B,e)$ or B or e, it follows (c.f. universal-introduction) that $B \models \forall(e \rightarrow Bot'(B,e))$. ∎

Lemma 3.2.6. *Given a clausal theory B and a clause e, let $Bot'(B,e)$ denote Generalised Bottom Set of B and e, then*

$$h \succeq Bot(B,e) \quad \textit{iff} \quad h \succeq Bot'(B,e)$$

Proof. [**Only-If-Part**]. If $h \succeq Bot'(B,e)$ then $h \succeq Bot(B,e)$ by transitivity, using $Bot'(B,e)\sigma \succeq Bot(B,e)$ from the argument above. [**If-Part**]. If $h \succeq Bot(B,e)$ then $h\theta \subseteq Bot(B,e)$ by definition of θ-subsumption. Now, since $h\theta$ is ground, and σ is a Skolemising substitution, it follows (c.f. Lemma A.7.2.3) that $h\theta = h\phi\sigma$ where $\phi = \{X/t' \mid X/t \in \theta \text{ and } t' = t\sigma^{-1}\}$. Hence $h\phi\sigma \subseteq Bot'(B,e)\sigma$. And since σ binds all of the variables in $Bot'(B,e)$ and $h\phi$ to Skolem distinct constants not appearing in $Bot'(B,e)$ or $h\phi$, it follows (c.f. Lemma A.7.2.2) that $h\phi \subseteq Bot'(B,e)$. Consequently $h \succeq Bot'(B,e)$. ∎

Proposition 3.2.7 (Correspondence of Bottom Generalisation and Relative Subsumption - ii). *Given a clausal theory B, a clause e, and a clause h, let σ denote the Skolemising substitution used in $Bot(B,e)$ and let $Bot'(B,e)$ denote Generalised Bottom Set of B and e. Then*

$$h \succeq Bot(B,e) \quad \textit{iff} \quad B \models \forall(e \leftrightarrow f) \textit{ and } h \succeq f \textit{ for some clause } f$$

Proof. [**Only-If-Part**]. If $h \succeq Bot(B,e)$ then $h \succeq Bot'(B,e)$ by Lemma 3.2.6, and $B \models \forall(e \leftrightarrow Bot'(B,e))$ by Lemma 3.2.5. [**If-Part**]. Given $B \models \forall(e \leftrightarrow f)$ and $h \succeq f$. Using

$B \models \forall(e \leftrightarrow f)$ it follows that $B \models \forall(f \leftrightarrow Bot'(B,e))$ by Lemma 3.2.5. Next, let γ be any grounding substitution for the formula $f\sigma$. Therefore $B \models f\sigma\gamma \leftrightarrow Bot(B,e)$ using elementary properties of substitutions. Now, either $Bot(B,e)$ is a tautology, or it is not. **Case 1**: If $Bot(B,e)$ is a tautology, then it must contain some ground atom A and its negation $\neg A$. Hence by Definition 16 it follows $B \cup \overline{e}$ entails both A and $\neg A$. Consequently $B \cup \overline{e}$ is inconsistent and entails all atoms and their negations. Therefore $Bot(B,e) = \blacksquare$ by Definition 16, and is subsumed by all hypotheses h, by Definition of θ-subsumption. **Case 2**: If $Bot(B,e)$ is not a tautology, then using $B \models f\sigma\gamma \leftrightarrow Bot(B,e)$ it follows $B \models Bot(B,e) \rightarrow f\sigma\gamma$ (c.f. implication-introduction) and $B \models \overline{Bot(B,e)} \rightarrow \overline{f\sigma\gamma}$ by taking the contrapositive. Therefore $B \cup \overline{Bot(B,e)} \models \overline{f\sigma\gamma}$. Now, because the RHS is a conjunction of ground atoms, and because every ground atom entailed by $B \cup \overline{Bot(B,e)}$ is already entailed by $\overline{Bot(B,e)}$ (observe that $\overline{Bot(B,e)}$ is the set of ground atoms entailed B and $\overline{e}$, so adding B does not result in any new ground atoms being entailed), it follows $\overline{Bot(B,e)} \models \overline{f\sigma\gamma}$. And because, by the case assumption, $Bot(B,e)$ is not a tautology, $\overline{Bot(B,e)}$ must be consistent, and so $\overline{f\sigma\gamma} \subseteq \overline{Bot(B,e)}$ (as this is the only way one consistent conjunction of ground atoms may entail another). Consequently $f\sigma\gamma \subseteq Bot(B,e)$ by negating elements, and therefore $f \succeq Bot(B,e)$ by definition of θ-subsumption. And finally $h \succeq Bot(B,e)$ by the transitivity of subsumption. ▮

As a final comment, note that the Generalised Bottom Set is not the only clause that will do. For example, the shorter clause $h\phi \cup e$ (obtained by collecting all of the literals in the clauses $h\phi$ and e) is also a suitable candidate for the clause f.

iii) Third Form of Relative Subsumption: $(B \cup \overline{e}, h) \vdash_c \Box$

The c-refutation formulation $(B \cup \overline{e}, h) \vdash_c \Box$ was shown in Proposition 2.3.2 to be equivalent to relative subsumption. Because of proximity of c-refutations to resolution-based procedures, they are perhaps the most suitable formulation in the context of automated induction. The equivalence of Bottom Generalisation to the c-refutation was established by Yamamoto, albeit under the additional assumption $B \not\models e$, which is dispensed with in Proposition 3.2.8 below.

Proposition 3.2.8 (Equivalence of Bottom Generalisation and Relative Subsumption - iii - [Yam99][11]). *Given a clausal theory B, a clause e, and a clause h containing no Skolem constants introduced in $Bot(B,e)$. Then*

$$h \succeq Bot(B,e) \quad \textit{iff} \quad (B \cup \overline{e}, h) \vdash_c \Box$$

Proof. From [Yam99] Thm4.1, Thm4.3 and Thm4.4, it can be deduced that under the assumption $B \not\models e$, it follows $h \succeq Bot(B,e)$ iff $(B \cup \overline{e}, h) \vdash_c \Box$. Now consider the case $B \models e$. Then on the one hand: $B \cup \overline{e}$ is inconsistent and all literals are contained in $Bot(B,e)$ which is therefore subsumed by every possible hypothesis h. And on the other hand: $B \cup \overline{e}$ is inconsistent and so there is a trivial c-derivation of $\Box$ that does not use h at all. Therefore the equivalence also holds when $B \models e$ (in which case both sides are true). ▮

[11]Proposition 3.2.8 is a generalisation of the result stated in [Yam97] Lem1, and demonstrated in [Yam99]. [Yam97] stated the two assumptions $e \not\equiv \top$ and $B \not\models e$, of the former is clearly a special case of the latter and may be dropped (i.e. insisting $B \not\models e$ already guarantees that $e \not\equiv \top$). [Yam99] stated the single assumption $B \not\models e$, which as shown in Proposition 3.2.8 above can also be dropped.

3.2.3 Vacuous Hypotheses and Literals

Having identified the hypotheses derivable by Bottom Generalisation, it is now argued that not all such hypotheses are practically useful or even intuitively satisfactory. The purpose of the following discussion is to motivate and formalise one particular class of undesirable hypotheses, which will be called the class of *vacuous hypotheses* on account of their being devoid of any useful content. The notion of a vacuous hypothesis will then be used to define the notion of a *vacuous literal* – that is to say a literal which is contained in the Bottom Set but would only lead to the formation of vacuous hypotheses were that literal to participate in the process of Bottom Generalisation. This notion will subsequently lead to a refinement of Bottom Generalisation that benefits from the non-computation of vacuous literals, and yet includes all of the hypotheses computed by Progol5.

Intuitively, there are two ways in which a hypothesis can be vacuous. First, a hypothesis is vacuous if it is not required to cover the seed example. Suppose, for example, that a learner is asked to explain why "everyone with blue eyes has blue eyes". The response "because pigs fly" is vacuous. This situation arises when $B \models e$, as in the previous example, where $B = \emptyset$ and $e = blueEyes(X)$:- $blueEyes(X)$ and $h = pigsFly$. Because there is nothing to explain, all hypotheses are superfluous. Second, a hypothesis is vacuous if it contradicts the premiss of the example. Suppose, for example, that a learner is asked to explain why "everyone with blue eyes has blonde hair". The response "because no one has blue eyes" is vacuous. This situation arises when $B \cup \{h\} \cup e^- \models \Box$ (recalling that e^- denotes the body atoms of e), as in the previous example, where $B = \emptyset$ and $e = blueHair(X)$:- $blueEyes(X)$ and $h =$:- $blueEyes(X)$.

But note there is one circumstance when it *is* necessary for h to contradict the body of e: when e is a negative clause and we are learning integrity constraints. Note also that hypotheses which entail the head of e without using the body have not been defined as vacuous. The justification is related to the consistency discussion of Section 3.1.2, but now considering positive rather then negative examples. It was suggested earlier that examples are usually obtained on the basis of observations. If the clause $a(X)$:- $b(X)$ is set as a positive example, it must be because all individuals known to have property b were observed to have property a. Hence it is reasonable to assume that at least one individual is known to have property b. For if not, then a general law would have been asserted as fact without the support of a single example! But it *is* conceivable that only those individuals with property b were tested for a, and so to extend property a to other individuals is not unreasonable.

The notion of a vacuous hypothesis is now formalised in Definition 20 below. Note that this definition could have be generalised to apply to inference methods other than Bottom Generalisation, by simply replacing the hypothesis clause h by a clausal theory H. The notion of vacuity is then extended to literals of the Bottom Set in Definition 21. Informally, a literal is vacuous if it can only participate in the construction of vacuous hypotheses. The motivation here is that if one does not wish to compute vacuous hypotheses, then a great deal of unnecessary work can by avoided by simply not computing vacuous literals. The remainder of this section will then show this how goal can be (partially) achieved with relative ease in a Horn clause setting, and the next section will reveal that this fact is exploited by the Progol proof procedure. But first, Example 3.2.9 provides a more concrete illustration of the concept a vacuous literal.

Definition 20 (Vacuous Hypothesis). Given a clausal theory B and a clause e, then a clause h derived by Bottom Generalisation from B and e is *vacuous* iff either of the following two conditions hold:

i. $B \models e$

ii. $B \cup \{h\} \cup e^- \models \Box$ and $e^+ \neq \emptyset$ ♦

Definition 21 (Vacuous Literal). Given a clausal theory B and a clause e. Let $L \in Bot(B,e)$ be a ground literal. Then L is **vacuous** iff every clause h is vacuous for which there exists a substitution θ such that $h\theta \subseteq Bot(B,e)$ and $L \in h\theta$. ♦

Example 3.2.9. Let B contain the negative clause $\text{:-}\, b, c$, and let e be the definite clause $a \,\text{:-}\, c$, where a, b and c are propositions. Observe that $Bot(B,e) = a, b \,\text{:-}\, c$ and that in total there are five hypotheses which can be derived from B and e by Bottom Generalisation: i) a, ii) $a \,\text{:-}\, c$, iii) b, iv) $b \,\text{:-}\, c$, and v) $\text{:-}\, c$. Verify that hypotheses iii, iv and v. are vacuous, but that i. and ii. are not. Note, therefore, that the literal b is vacuous, but that a and c are not.

3.2.4 Standardisation Procedure for Horn Clauses

In order to automate the processes of inductive learning, effective procedures are required for deciding entailment between theories and clauses. Such procedures are required, for example, in the computation of literals in the Bottom Set, and in the testing of coverage and consistency. Deciding entailment is highly intractable in full clausal logic, but is considerably easier for definite clause logic – where efficient logic programming tools have been developed. In practice, definite clause logic is not sufficiently expressive for many applications, and so Horn clause logic is considered to be more suitable. In order to efficiently support Horn clause logic, many systems, including Progol, make use of a simple transformation that allows Horn entailment to be decided using conventional logic programming methods. One such transformation, called *Standardise* is formalised in Definition 22 and analysed below.

Definition 22 (Standardise). Given a Horn theory B and a Horn clause e, let ff be a proposition symbol occurring in neither B nor e, and let σ be a Skolemising substitution for both B and e, then the *Standardisation of B and e*, denoted $Standardise(B,e)$, is the pair $\langle \mathcal{B}, \epsilon \rangle$ defined as follows:

$$Standardise(B,e) = \langle \mathcal{B}, \epsilon \rangle \text{ where } \mathcal{B} = B_{\vec{\mathsf{ff}}} \cup e^-_\sigma \text{ and } \epsilon = (e_{\vec{\mathsf{ff}}})^+_\sigma$$ ♦

Informally, *Standardise* rearranges the given background B and the given example e to output a standardised background $\mathcal{B}$ and a standardised example ϵ, that are more amenable to logic programming techniques. To understand this transformation, recall from Definition 2 that the incapitation $T_{\vec{\mathsf{ff}}}$ inserts the proposition ff into the head of every negative clause in theory T. Recall from Section 2.1 that the Skolemising substitution C_σ replaces every variable in clause C by a fresh Skolem constant, and that C^+ and C^- denote the head and body atoms of C. The transformation can now be understood thus. First, *Standardise* incapitates the (Horn) theory B and clause e to give the (definite) theory $B_{\vec{\mathsf{ff}}}$ and clause $e_{\vec{\mathsf{ff}}}$. Then, the Skolemising substitution σ is applied to the latter, giving the (ground) clause $(e_{\vec{\mathsf{ff}}})_\sigma$. Finally, the head (atom) $(e_{\vec{\mathsf{ff}}})^+_\sigma$ of this clause becomes the standardised example ϵ, and the body atoms $(e_{\vec{\mathsf{ff}}})^-_\sigma$ or simply e^-_σ are added to $B_{\vec{\mathsf{ff}}}$ to give the standardised background $\mathcal{B}$.

Example 3.2.10. If the given background knowledge and given example are

$$B = \left\{ \begin{array}{l} r(X)\,\text{:-}\,q(X) \\ \text{:-}\,p(X) \end{array} \right\} \qquad e = p(X)\,\text{:-}\,q(X)$$

then the standardised background knowledge and standardised example are

$$\mathcal{B} = \left\{ \begin{array}{l} r(X)\,\text{:-}\,q(X) \\ \mathsf{ff}\,\text{:-}\,p(X) \\ q(k) \end{array} \right\} \qquad \epsilon = p(k)$$

where k is a fresh Skolem constant.

Observe, that the *Standardise* transformation always returns a definite theory $\mathcal{B}$ and a ground atom e. The importance of this transformation is now established in Proposition 3.2.11 below, where it is shown how *Standardise* enables entailment between Horn clauses to be decided by SLD procedures. If $\mathcal{B}$ and ϵ represent the standardised counterparts of B and e then there are two possibilities: either $\mathcal{B} \vdash ?\mathsf{ff}$, corresponding to the case e is trivially implied because its body is inconsistent with B; or else $\mathcal{B} \vdash ?\epsilon$, or in other words, when the query $?\epsilon$ succeeds from the program $\mathcal{B}$ by SLD-resolution.

Proposition 3.2.11 (Deciding Entailment via Standardisation). *Given a Horn theory B and a Horn clause e, let $\mathcal{B}$ and ϵ denote the standardisation of B and e. Then*

$$B \models e \ \text{ iff } \ \mathcal{B} \vdash ?\epsilon \text{ or } \mathcal{B} \vdash ?\mathit{ff}$$

Proof. [**Only-If-Part**]. If $B \models e$ then by the entailment theorem $B \wedge \neg e \models \bot$ and since $\bot$ does not contain any Skolem symbols $B \wedge \overline{e_\sigma} \models \bot$ from Definition 1. **Case 1**: e is a definite clause. Writing the LHS explicitly $B \wedge \neg E_0\sigma \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \bot$ so by the entailment theorem $B \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models E_0\sigma$. Now by definition of clauses and substitutions $\{E_1\sigma, \ldots, E_n\sigma\} = e_\sigma^-$ and because e is a definite clause $E_0\sigma = \epsilon$. Therefore $B \cup e_\sigma^- \models \epsilon$. Now, either i) $B \cup e_\sigma^- \models \bot$ so that $B_{\vec{\mathsf{ff}}} \cup e_\sigma^- \models \mathsf{ff}$ by Definition 2 so that $B_{\vec{\mathsf{ff}}} \cup e_\sigma^- \vdash ?\mathsf{ff}$ by the completeness of SLD for definite theories, and $\mathcal{B} \vdash ?\mathsf{ff}$ by Definition 22. Or else, ii) $B \cup e_\sigma^- \vdash ?\epsilon$ by the completeness of SLD resolution for Horn theories. And since only definite clauses in B may participate in the derivation of the query ϵ it follows $B_{\vec{\mathsf{ff}}} \cup e_\sigma^- \vdash ?\epsilon$ and therefore $\mathcal{B} \vdash ?\epsilon$. **Case 2**: e is a negative clause. Writing the LHS explicitly $B \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \bot$ and so $B \cup e_\sigma^- \models \bot$ and $\mathcal{B} \vdash ?\mathsf{ff}$ as in case 1.

[**If-Part**]. **Case 1**: $\mathcal{B} \vdash ?\mathsf{ff}$. Then $B_{\vec{\mathsf{ff}}} \cup e_\sigma^- \vdash ?\mathsf{ff}$ by Definition 22 and $B_{\vec{\mathsf{ff}}} \cup e_\sigma^- \models \mathsf{ff}$ by the soundness of SLD. Therefore $B \cup e_\sigma^- \models \bot$ by Definition 2 and writing the LHS explicitly $B \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \bot$. Now, either e is negative, in which case $B \wedge \overline{e_\sigma} \models \bot$, or else e is definite, in which case $B \wedge \neg E_0\sigma \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \bot$ by monotonicity and again $B \wedge \overline{e_\sigma} \models \bot$. Either way $B \wedge \neg e \models \bot$ by Definition 2 and therefore $B \models e$ by the entailment theorem. **Case 2**: $\mathcal{B} \vdash ?\epsilon$. If e is negative then $\mathcal{B} \vdash ?\mathsf{ff}$ by Definition 22 and $B \models e$ as in case 1. If e is definite then $B_{\vec{\mathsf{ff}}} \cup e_\sigma^- \vdash ?E_0\sigma$ by Definition 22 and $B_{\vec{\mathsf{ff}}} \cup e_\sigma^- \models E_0\sigma$ by the soundness of SLD. Writing the LHS explicitly $B \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models E_0\sigma$. Hence $B \wedge \neg E_0\sigma \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \bot$ by the entailment theorem, and so $B \models e$ as in case 1. ∎

The *Standardise* transformation therefore provides a general method for deciding entailment between a Horn theory and a Horn clauses using SLD. As a special case, by setting $e = \square$, this same procedure allows the consistency of a Horn theory B to be decided. As a simple extension, the coverage of a set of examples E may be decided by testing each individual example in turn. Together, these observations motivate Definitions 23 and 24 below, which can in turn be used as the basis of revised definitions of coverage, consistency and compression.

Definition 23 (Covers).

$$Covers(B,e) \;\equiv\; \mathcal{B} \vdash ?\epsilon \text{ or } \mathcal{B} \vdash ?\mathsf{ff} \text{ where } \langle \mathcal{B}, \epsilon \rangle = Standardise(B,e) \qquad \blacklozenge$$

Definition 24 (Consistent).

$$Consistent(B) \;\equiv\; \neg Covers(B, \Box) \qquad \blacklozenge$$

3.2.5 Refined Bottom Generalisation

The standardisation procedure described above is now used to obtain a refinement of the Bottom Set that excludes a certain class of vacuous literals. This refinement is defined for Horn clause logic, but uses the *Standardise* transform to exploit standard logic programming techniques. In order to show how the notion of *Refined Bottom Generalisation* is derived, it is convenient to analyse separately the body and head atoms of the Bottom Set, as done in Lemmas 3.2.13 and 3.2.14 below. In both cases a disjunctive characterisation is obtained in terms of the standardisation $\mathcal{B}$ and ϵ of the inputs B and e. It is then shown in Lemmas 3.2.15 and 3.2.16 that the effect of dropping the second of the two disjuncts results only in the exclusion of vacuous literals. This fact is used in Definitions 25 and 26 to define the concepts of *Refined Bottom Set* and *Refined Bottom Generalisation.*

First, Lemma 3.2.12 below, establishes the fact that the atom ff appears in the head and body of Bottom Set if and only if B entails e, or equivalently, if the Bottom Set is the null-clause $\blacksquare$. Recall from Section 2.1 that the null-clause is the clause that contains all ground literals and is θ-subsumed by all other clauses.

Lemma 3.2.12 (Degeneracy of Bottom Set). *Given a Horn theory B and a Horn clause e, then*

$$B \models e \;\; \textit{iff} \;\; Bot(B,e) = \blacksquare \;\; \textit{iff} \;\; ff \in Bot^{+}(B,e) \;\; \textit{iff} \;\; ff \in Bot^{-}(B,e)$$

Proof. If $B \models e$ then $B \wedge \neg e \models \bot$ by the entailment theorem. Hence $B \wedge \overline{e} \models \bot$ by Definition 1. And so for any atom A it follows $B \cup \overline{e}$ entails both A and its negation $\neg A$. Hence $Bot(B,e) = \blacksquare$ by Definition 16. In particular $Bot(B,e)$ contains the atom ff and its negation $\neg$ff, hence ff $\in Bot^{+}(B,e)$ and ff $\in Bot^{-}(B,e)$ by definition of the head of a clause. If on the other hand ff $\in Bot^{+}(B,e)$ or ff $\in Bot^{-}(B,e)$ by Definition 16 it follows $B \cup \overline{e}$ entails ff or its negation $\neg$ff. And since, by assumption, the proposition ff is not mentioned at all in $B \cup \overline{e}$, it must be that $B \wedge \overline{e} \models \bot$ and so $B \wedge \neg e \models \bot$ by Definition 1. Therefore $B \models e$ by the entailment theorem. ▮

Lemmas 3.2.13 and 3.2.14 now characterise the body and head literals of the Bottom Set in terms of the standardised background knowledge and example.

Lemma 3.2.13 (Body of Bottom Set). *Given a Horn theory B, a Horn clause e, and a ground atom $\delta \neq ff$, let $\langle \mathcal{B}, \epsilon \rangle = Standardise(B,e)$. Then*

$$\delta \in Bot^{-}(B,e) \;\; \textit{iff} \;\; \mathcal{B} \vdash ?\delta \; \textit{or} \; B \models e$$

Proof. [**Only-If-Part**]. If $\delta \in \mathit{Bot}^-(B,e)$ then $B \cup \overline{e_\sigma} \models \delta$ by Definition 16. Now, either i) $B \cup \overline{e_\sigma} \models \bot$, so that $B \models e$ as in Lemma 3.2.12. Or else ii) $B \cup \overline{e_\sigma} \vdash ?\delta$ by the completeness of SLD for Horn clauses. Now, writing the LHS explicitly $B \cup \{\neg E_0\sigma, E_1\sigma, \ldots, E_n\sigma\} \vdash ?\delta$, where $\neg E_0\sigma$ may or may not exist (depending on whether e is definite or negative). But whether it exists or not, it follows that $B_{\vec{\mathsf{ff}}} \cup \{E_1\sigma, \ldots, E_n\sigma\} \vdash ?\delta$ by Definition 2 (n.b.b. use fact no negative clauses on the LHS many participate in derivation of δ). But this can be written $B_{\vec{\mathsf{ff}}} \cup e_\sigma^- \vdash ?\delta$, and therefore $\mathcal{B} \vdash ?\delta$ by definition 22.

[**If-Part**]. **Case 1**: If $\mathcal{B} \vdash ?\delta$ then $\mathcal{B} \models \delta$ by the soundness of SLD. And so $B_{\vec{\mathsf{ff}}} \cup \{E_1\sigma, \ldots, E_n\sigma\} \models \delta$ by Definition 22, and hence $B_{\vec{\mathsf{ff}}} \cup \{\neg\mathsf{ff}\} \cup \{E_1\sigma, \ldots, E_n\sigma\} \models \delta$ by monotonicity. Therefore $B \cup \{E_1\sigma, \ldots, E_n\sigma\} \models \delta$ by Definition 2, and so $B \cup \{\neg E_0\sigma, E_1\sigma, \ldots, E_n\sigma\} \models \delta$ by monotonicity. Therefore $B \cup \overline{e} \models \delta$ and $\delta \in \mathit{Bot}^-(B,e)$ by Definition 16. **Case 2**: If $B \models e$ then $\mathit{Bot}(B,e) = \blacksquare$ by Lemma 3.2.12, and so $\delta \in \mathit{Bot}^-(B,e)$ by Definition of the null-clause. ▮

Lemma 3.2.14 (Head of Bottom Set). *Given a Horn theory B and a Horn clause e, and a ground atom $\alpha \neq ff$. Let $\langle \mathcal{B}, \epsilon \rangle = \mathit{Standardise}(B,e)$. Then*

$$\alpha \in \mathit{Bot}^+(B,e) \quad \textit{iff} \quad \mathcal{B} \cup \{\alpha\} \vdash ?\epsilon \ \textit{or} \ \mathcal{B} \cup \{\alpha\} \vdash ?\mathit{ff}$$

Proof. [**Only-If-Part**]. If $\alpha \in \mathit{Bot}^+(B,e)$ then by Definition 16 it follows $B \wedge \overline{e_\sigma} \models \neg\alpha$, and so $B \wedge \alpha \wedge \overline{e_\sigma} \models \bot$ by the entailment theorem. **Case 1**: e is a definite clause. Writing the LHS explicitly $B \wedge \alpha \wedge \neg E_0\sigma \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \bot$. Therefore $B \wedge \alpha \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models E_0\sigma$ by the entailment theorem. Now, either i) $B \wedge \alpha \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \bot$ and $B_{\vec{\mathsf{ff}}} \wedge \alpha \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \mathsf{ff}$ by Definition 2, which is equivalent to $\mathcal{B} \cup \alpha \vdash ?\mathsf{ff}$. Or else ii) $B \wedge \alpha \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \vdash ?E_0\sigma$ by the completeness of SLD for Horn clauses. Therefore it follows that $B_{\vec{\mathsf{ff}}} \wedge \alpha \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \vdash ?E_0\sigma$ (n.b.b. either use fact no negative clauses on the LHS many participate in derivation of $E_0\sigma$; alternatively use fact incapitation leaves unchanged the consequences of a consistent theory. n.b the latter requires $\alpha \neq \mathsf{ff}$ so incapitation is defined on the LHS; the former does not). And this can be written $\mathcal{B} \cup \{\alpha\} \vdash ?\epsilon$. **Case 2**: e is a negative clause. Writing the LHS explicitly $B \wedge \alpha \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \bot$ and $\mathcal{B} \cup \{\alpha\} \vdash ?\mathsf{ff}$ as in Case 1 part ii. This time $\epsilon = \mathsf{ff}$ and so in addition $\mathcal{B} \cup \{\alpha\} \vdash ?\epsilon$. [**If-Part**]. **Case 1**: $\mathcal{B} \cup \{\alpha\} \vdash ?\mathsf{ff}$. Then $\mathcal{B} \cup \{\alpha\} \models \mathsf{ff}$ by the soundness of SLD and $B_{\vec{\mathsf{ff}}} \cup \{E_1\sigma, \ldots, E_n\sigma\} \cup \{\alpha\} \models \mathsf{ff}$ by Definition 16. Therefore $B \cup \{E_1\sigma, \ldots, E_n\sigma\} \cup \{\alpha\} \models \bot$ by Definition 2. Consequently $B \cup \{\neg E_0\sigma, E_1\sigma, \ldots, E_n\sigma\} \cup \{\alpha\} \models \bot$ by monotonicity and $B \cup \{\neg E_0\sigma, E_1\sigma, \ldots, E_n\sigma\} \models \neg\alpha$ by the entailment theorem. Therefore $B \cup \overline{e} \models \neg\alpha$ by Definition 1 and so $\alpha \in \mathit{Bot}^+(B,e)$ by Definition 16. **Case 2**: $\mathcal{B} \cup \{\alpha\} \vdash ?\epsilon$. Now, either e is negative and so $\epsilon = \mathsf{ff}$ and $\alpha \in \mathit{Bot}^+(B,e)$ as in case 1. Or else, e is definite and so $\epsilon = E_0\sigma$. In which case $\mathcal{B} \cup \{\alpha\} \models E_0\sigma$ by the soundness of SLD and $B_{\vec{\mathsf{ff}}} \cup \{E_1\sigma, \ldots, E_n\sigma\} \cup \{\alpha\} \models E_0\sigma$ by Definition 16. Consequently $B_{\vec{\mathsf{ff}}} \cup \{\neg\mathsf{ff}\} \cup \{E_1\sigma, \ldots, E_n\sigma\} \cup \{\alpha\} \models E_0\sigma$ by monotonicity and $B \cup \{E_1\sigma, \ldots, E_n\sigma\} \cup \{\alpha\} \models E_0\sigma$ by Definition 2. Therefore $B \cup \{\neg E_0\sigma, E_1\sigma, \ldots, E_n\sigma\} \cup \{\alpha\} \models \neg\alpha$ by the principle of IE. Hence $B \cup \overline{e} \models \neg\alpha$ by Definition 1 and so $\alpha \in \mathit{Bot}^+(B,e)$ by Definition 16. ▮

Note that the requirement $\delta = \mathsf{ff}$ can be dropped from Lemma 3.2.13, as the LHS is equivalent to $B \models e$ using Lemma 3.2.12 (first and last formulae), and RHS is equivalent to $B \models e$ using Proposition 3.2.11. Note that the requirement $\alpha = \mathsf{ff}$ cannot be dropped from Lemma 3.2.14, as the LHS is equivalent to $B \models e$ using Lemma 3.2.12, but the RHS is equivalent to $\top$ (by second disjunct). Note, however, rather than choosing to exclude the atom ff from $\mathit{Bot}(B,e)$, it will be included whenever e is a negative clause. If selected to appear in the head of a hypothesis, it will later be removed to yield an integrity constraint.

Lemmas 3.2.15 and 3.2.16 below now consider the effect of dropping the second of the two disjuncts on the RHS of Lemmas 3.2.13 and 3.2.14 above. Clearly, dropping one of the criteria by which literals may gain entry to the Bottom Set, will result in the loss of those literals which satisfy that criterion, but not the other. In both cases, this is shown that the only literals thus excluded, are all vacuous.

Proposition 3.2.15 (Elimination of Vacuous Body Atoms). *Given a Horn theory B, a Horn clause e, and a ground atom $\delta \in Bot^-(B,e)$, let $\langle \mathcal{B}, \epsilon \rangle = Standardise(B,e)$. Then*

$$B \models e \textit{ and } \mathcal{B} \nvdash ?\delta \textit{ implies } \delta \textit{ is vacuous}$$

Proof. If $B \models e$ then all hypotheses derivable by Bottom Generalisation from B and e are vacuous by Definition 20. Therefore all of those hypotheses that 'use' δ are vacuous (as are all those that do not). And so the literal δ is vacuous by Definition 21. ∎

Proposition 3.2.16 (Elimination of Vacuous Head Atoms). *Given a Horn theory B, a Horn clause e, and a ground atom $\alpha \in Bot^+(B,e)$, let $\langle \mathcal{B}, \epsilon \rangle = Standardise(B,e)$. Then*

$$\mathcal{B} \cup \{\alpha\} \vdash ?\mathsf{ff} \textit{ and } \mathcal{B} \cup \{\alpha\} \nvdash ?\epsilon \textit{ implies } \alpha \textit{ is vacuous}$$

Proof. If $\mathcal{B} \cup \{\alpha\} \vdash ?\mathsf{ff}$ then $B_{\overline{\mathsf{ff}}} \cup e_\sigma^- \cup \{\alpha\} \vdash ?\mathsf{ff}$ by Definition 22 and so $B_{\overline{\mathsf{ff}}} \cup e_\sigma^- \cup \{\alpha\} \models \mathsf{ff}$ by the soundness of SLD resolution. Then, either i) $\alpha = \mathsf{ff}$ in which case $B \models e$ by Lemma 3.2.12, and α is vacuous by Definition 20. Or else ii) $\alpha \neq \mathsf{ff}$ and so $B \cup e_\sigma^- \cup \{\alpha\} \models \bot$ by Definition 2. In this case $B \cup e_\sigma^- \models \neg\alpha$ by the entailment theorem. Now, let $h = A \text{:-} D_1, \ldots, D_n$ be any Horn clause such that for some substitution θ it is the case $h\theta \doteq \alpha \text{:-} \delta_1, \ldots, \delta_n$ and $\delta_i \in Bot^-(B,e)$ for all $1 \leq j \leq n$. (In other words, h is any hypothesis derivable by Bottom Generalisation that 'uses' the literal α). Then by Lemma 3.2.13 it follows that either a) $B \models e$, in which case α is vacuous by Definition 20. Or else b) $\mathcal{B} \models \delta_i$. But then, if $\delta_i = \mathsf{ff}$ for some $1 \leq j \leq n$ then again $B \models e$ by Lemma 3.2.12 and α is vacuous by Definition 20. And similarly, if $\delta_i \neq \mathsf{ff}$ for all $1 \leq j \leq n$, then from $\mathcal{B} \models \delta_i$ it follows $B_{\overline{\mathsf{ff}}} \cup e_\sigma^- \models \delta_i$ by Definition 22, and using $\delta_i \neq \mathsf{ff}$ it then follows $B \cup e_\sigma^- \models \delta_i$ by Definition 2 (n.b.b. use subsumption theorem and fact that no incapitated clauses may participate in derivation). Hence $B \cup e_\sigma^- \models \neg\alpha \wedge \delta_1 \wedge \ldots \wedge \delta_n$ (c.f. and-introduction using facts that each δ_i is entailed individually, and so is $\neg\alpha$ as α is in the head of the Bottom Set), which is equivalent to $B \cup e_\sigma^- \models \neg h\theta$. Because $h \succcurlyeq h\theta$ and $h \models h\theta$, both by Definition of θ-subsumption, it follows o $\neg h\theta \models \neg h$ by the principle if IE. Therefore $B \cup e_\sigma^- \models \neg h$ by transitivity of entailment, and hence $B \cup \{h\} \cup e_\sigma^- \models \bot$ by the entailment theorem. And since $\mathcal{B} \cup \{\alpha\} \vdash ?\mathsf{ff}$ and $\mathcal{B} \cup \{\alpha\} \nvdash ?\epsilon$ imply that $\epsilon \neq \mathsf{ff}$ and hence $e^+ \neq \emptyset$, it follows that h is vacuous by Definition 20. And since h was chosen arbitrarily it follows that all such hypotheses, for which there exists a substitution θ with the above properties, are vacuous (c.f. universal introduction). Hence the literal α is vacuous by Definition 21. ∎

The preceding results motivate the formulation of Refined Bottom Generalisation shown in Definition 25 below. The Refined Bottom Set differs from the standard Bottom Set in only two respects: i) it excludes a (sub)set of head and body atoms shown above to be vacuous, and ii) it includes the atom ff whenever e is a negative clause. Note that the notation $\{\alpha_1, \ldots, \alpha_n\} \text{:-} \{\delta_1, \ldots, \delta_n\}$ is intended to represent the clause $\alpha_1, \ldots, \alpha_n \text{:-} \delta_1, \ldots, \delta_n$.

Definition 25 (Refined Bottom Set). Given a Horn theory B and a Horn clause e such that $B \not\models e$. Let $\langle \mathcal{B}, \epsilon \rangle = \mathit{Standardise}(B, e)$. Then

$$Bot^*(B,e) = \{\alpha \in \mathcal{GA} \mid \mathcal{B} \wedge \alpha \vdash ?\epsilon\} \mathrel{:-} \{\delta \in \mathcal{GA} \mid \mathcal{B} \vdash ?\delta\} \qquad \blacklozenge$$

Definition 26 (Refined Bottom Generalisation). Given a Horn theory B and a Horn clause e such that $B \not\models e$, let $\langle \mathcal{B}, \epsilon \rangle = \mathit{Standardise}(B, e)$, and let h be any clause containing no Skolem constant nor the proposition symbol ff. Then h is *derivable by Refined Bottom Generalisation from B and e* iff

$$h_{\vec{\mathsf{ff}}} \succcurlyeq Bot^*(B,e) \qquad \blacklozenge$$

What is important about the definitions above, whether one agrees fully with the philosophy that led to their formalisation, is the Progol5 proof procedure will be seen in the next section to compute only a subset of the Refined Bottom Set. For the benefit of the curious reader, it is simply stated here that: i) the body atoms excluded from the Refined Bottom Set are not computed by the Progol5 BOTTOMSET routine on account of the cover-removal step in the Progol5 COVERSET loop, ii) the excluded head atoms are not computed because the Progol5 STARTSET routine only adds the contrapositive ff^* in the case that e is a negative clause. Note also that if any hypotheses were to be considered that contained a Skolem constant, it would be eliminated by the Progol5 SEARCH routine. For further details see Section 3.3. But first, this section concludes by introducing a refinement of the MSH cover-set approach, motivated by the above definitions.

3.2.6 Bottom-Based Cover-Set Generalisation

The motivation for combining the cover-set-approach and the inference method of Bottom Generalisation can be understood in two ways. First, the cover-set-approach can be seen as a mechanism by which Bottom Generalisation, which treats single seed examples, can be applied to the task of ILP, which involves multiple positive and negative examples. Second, Bottom Generalisation can be seen as a concrete inference method which can be used to implement the cover-set approach. Since these are two sides of the same coin, so to speak, only one need be developed, and the latter is chosen below.

If automated ILP procedures are to be developed based on the approach of MSH Generalisation, then concrete methods are required for the construction and generalisation of MSH. In general both tasks are intractable, but under certain restrictions efficient procedures can be obtained. The Bottom Set construct arises naturally in this context when the generality ordering is taken as θ-subsumption and the MSH is restricted to a single ground clause. For under these conditions the Most Specific Hypothesis coincides with the Bottom Set. Moreover, the approach of Bottom Generalisation arises when the Most Compressive Generalisation is also restricted to a single clause. Furthermore, if the representation is restricted to Horn clauses, then the inference method Refined Bottom Generalisation can be used instead, and the *Standardise* transformation used to implement the testing of coverage and consistency. However, before this method can be automated, one significant obstacle remains: in general $Bot^*(B,e)$ is infinite and cannot be computed in its entirety. Fortunately, it is easily shown that considering only finite subsets of $Bot^*(B,e)$ results in a sound inference procedure.

Corollary 3.2.17 below is a straightforward consequence of the preceding results, and states the obvious but crucial fact that Bottom Generalisation can be realised by computing only a finite subset of the Bottom Set. The remainder of this Section is concerned with the

```
Begin Algorithm 3.2.18 (Bottom-COVERSET).
  Input              given a Horn context $\langle B, E^+, E^-, M\rangle$
                       assert $Consistent(B \cup E^+ \cup E^-)$
                       let $E^+ = E^+ - Cover(B, E^+)$
  Cover-Set-Loop       while $E^+ \neq \emptyset$
  select seed              choose seed example $e \in E^+$
  standardise              let $\langle \mathcal{B}, \epsilon\rangle = Standardise(B, e)$
  construct MSH            find $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$ such that $\mathcal{B} \cup \{\alpha_i\} \vdash ?\epsilon$ for all $1 \leq i \leq n$
  head                         for each $\alpha_i \in \mathcal{A}$
  body                             find $\beta_i = \alpha_i$ :- $\delta_1, \ldots, \delta_m$ such that $\mathcal{B} \vdash ?\delta_j$ for all $1 \leq j \leq m$
  generalise MSH                   find most compressive $h_i \in \mathcal{H}_M$ such that $h_i \succeq \beta_i$
  best hypothesis          let $h$ be the $h_i$ with greatest Compression
                           if $Compression(h, B, E^+, E^-) = -\infty$ then abort
  add hypothesis           let $B = B \cup \{h\}$
  remove cover             let $E^+ = E^+ - Cover(B, E^+)$
  Output             return $B$
End Algorithm (Bottom-COVERSET)
```

strategy used by Progol to turn this result effective procedure for efficiently realising Bottom Generalisation within Horn clause logic.

Corollary 3.2.17. *Let $\mathcal{B}$ and ϵ be the result of standardising a Horn theory B and a Horn clause e. If $\beta = \alpha$:- $\delta_1, \ldots, \delta_m$ is any (finite) definite clause such that: i) β contains no Skolem constant nor the proposition symbol ff, and ii) α is a ground atom such that $\mathcal{B} \cup \{\alpha\} \vdash ?\epsilon$, and iii) for all $1 \leq j \leq m$ it is the case that δ_j is a ground atom such that $\mathcal{B} \vdash ?\delta_j$. Then $h_{\bar{\mathrm{ff}}} \succeq \beta$ implies that h is derivable by Refined Bottom Generalisation from B and e*

Proof. If $h_{\bar{\mathrm{ff}}} \succeq \beta$, then since $h \succeq h_{\bar{\mathrm{ff}}}$ by definition 2, and since $\beta \succeq Bot(B, e)$ using the fact $\beta \subseteq Bot^*(B, e)$ by Definition 25, then $h \succeq Bot^*(B, e)$ by transitivity of θ-subsumption, and so h is derivable by Refined Bottom Generalisation from B and e by Definition 25. ∎

Corollary 3.2.17 motivates a Bottom-based cover-set methodology, which, as shown in Algorithm 3.2.18, refines Algorithm 3.1.7 in the following ways: First, the inputs and outputs are restricted to Horn clauses. Second, the new function *Consistent* has been introduced. Third, the Bottom Set replaces the MSH, and θ-subsumption replaces entailment as the generalisation relation. Fourth, the *Standardise* transform is applied immediately after seed selection to facilitate the construction of the Bottom Set. Fifth, the MCG output on each iteration of the Cover-Set Loop is restricted to a single Horn clause.

But note that instead of constructing a single Bottom Set on each iteration, several *candidate* MSH are instead constructed and generalised individually (a strategy used by Progol) - the most compressive such generalisation being returned as the MCG. Bottom Set construction is performed in two steps. The head atom is first chosen and then the body atoms are added. In practice the computation of body atoms is likely to depend on the head atom, in a fashion that will be guided by the language bias. Finally note that concrete procedures for selecting the literals *alpha* and δ_j and for performing the generalisation will be specified in the next section, which introduces the Progol proof procedure.

3.3 A Critical Analysis of the Progol Proof Procedure

This section provides a detailed analysis of the Progol5 proof procedure, which is a concrete refinement of the Bottom-based MSH procedure developed in the previous section. First the Progol4 technique of Mode Directed Inverse Entailment (MDIE) [Mug95] is described in Subsection 3.3.1. Then the Progol5 extension of Theory Completion with Inverse Entailment (TCIE) [MB00] is detailed in Subsection 3.3.2. Then, a mathematical treatment of the STARTSET Algorithm in Subsections 3.3.3 and 3.3.4, confirms its suspected soundness, but reveals a previously unsuspected source of incompleteness with respect to the semantics of Bottom Generalisation. Finally, the computation of vacuous literals is reconsidered in Subsection 3.3.5.

3.3.1 Mode Directed Inverse Entailment and Progol4

MDIE is a concrete instance of the Bottom-based Cover-Set Generalisation methodology described in Algorithm 3.2.18. The term 'Mode Directed' refers to the extensive use made of user-supplied mode-decs in both the construction and generalisation of MSH. The techniques of MDIE were first implemented in the ILP system Progol4, whose operation is described in Algorithm 3.3.1.

Progol4 takes as input the Horn clause theories B, E^+, E^- and a set of mode-decs M. The output is a Horn theory $B \cup H$ where $H = \{h_1, \dots, h_n\}$ is a valid inductive generalisation, and each h_i is added to B on successive iterations of the Cover-Set Loop. The tasks of MSH construction and generalisation are performed by two sub-routines called BOTTOMSET and SEARCH.

On each iteration of the Cover-Set Loop, a seed example e is first chosen from among the remaining positive examples E^+. Then standardised counterparts $\mathcal{B}$ and ϵ are obtained for the current background knowledge B and the current seed example e . Next, guided by the user-supplied mode-decs, one or more candidate MSH β_i are constructed by BOTTOMSET and individually generalised by SEARCH to give one or more candidate sub-hypotheses h_i. Finally the most compressive of these hypotheses is then asserted and any covered examples are removed. Each MSH β_i is a maximally specific Horn clause that θ-subsumes $Bot(B, e)$ and is constructible within the available language and search bias. Each β_i is built by BOTTOMSET one body atom at a time. Each hypothesis h_i is a maximally compressive Horn clause that θ-subsumes its corresponding β_i and is computed by STARTSET by means of a general-to-specific A^* search through the θ-subsumption sub-lattice $[\square, \beta_i]$. The SEARCH algorithm employed by Progol is described in detail in [Mug95] and will not be discussed further.

BOTTOMSET is the routine responsible for the computation of body atoms. As shown in Algorithm 3.3.3, the inputs are a definite theory $\mathcal{B}$, a ground atom ϵ, a head-dec m whose schema θ-subsumes ϵ, and a set of body-decs M^-. The output is a clause β such that $\beta \succcurlyeq Bot(B, e)$ and $\beta \in \mathcal{H}_{M^- \cup \{m\}}$. The routine also assumes a function *hash* that maps distinct constant symbols to distinct variables, and a parameter N_d called the *depth*.

BOTTOMSET begins by asserting that the schema a of the head-dec m does actually θ-subsume the atom ϵ, and, in so doing, it determines which terms in ϵ correspond to +types, −types and ♯types. To ensure compatibility with m, substitutions of the form $\{V/hash(t)\}$ ensure that each +type and −type variable in a is replaced by a suitable variable $hash(t)$,

Begin Algorithm 3.3.1 (Progol4 - [Mug95, MF01][12]).

```
Input              given $B$, $E^+$, $E^-$, $M$
check consist.       assert $Consistent(B \cup E^+ \cup E^-)$
remove cover         let $E^+ = E^+ - Cover(B, E^+)$
Cover-Set Loop       while $E^+ \neq \emptyset$
select seed              choose seed example $e \in E^+$
standardise              let $\langle \mathcal{B}, \epsilon \rangle = Standardise(B, e)$
select head-dec          for each head-dec $m_i \in M^+$ such that $m_i \succcurlyeq \epsilon$
bottom set                   let $\beta_i = B\text{OTTOM}S\text{ET}(\mathcal{B}, \epsilon, m_i, M^-)$
search                       let $h_i = S\text{EARCH}(\beta_i, B, E^+, E^-, M)$
assert best hyp.         add to $B$ the $h_i$ with greatest $Compression$
remove cover             let $E^+ = E^+ - Cover(B, E^+)$
Output             return $B$
```

End Algorithm (Progol4)

and each ♯type is replaced by the corresponding term t. Any terms bound by a +type are added to the set $InTerms$ and the head atom a is added to β. Note that different head-decs will in general lead to different patterns of term propagation and hence to different β_i.

Next begins the task of computing body atoms of beginning at a depth of 1 and successively increasing up to a maximum depth of N_D. The set $InTerms$ contains all terms corresponding to +types in the head atom, and −types in preceding body atoms of a strictly lesser depth. The set $NextTerms$ contains all terms corresponding to −types in preceding body at the current depth. New body atoms are decided by the successful instances under SLD of all possible queries obtained by substituting terms from $InTerms$ into the +types of the schemas all body-decs in M^-. Once again, constants are variablised where necessary in order to ensure compatibility with M^-. Type calls are added to ensure that at the ground level all constants are of the correct type.

Example 3.3.2. Referring to Figure 3.1, let $B = K_B$ as shown in Figure 2.1, let $\epsilon = fries(md)$, let $m = \mathsf{modeh}[fries(+bistro)]$, let $M^- = \{\mathsf{modeb}[offer(+bistro)]\}$, and assume that $hash(md) = Z$. Since $fries(V) \succcurlyeq fries(md)$ with the +type V binding to the constant md, the literal $fries(Z)$ is added to β and the constant md is added to $InTerms$ (both initially empty). After replacing the only +type W in $offer(W)$ by the only term md in $InTerms$, the query $?offer(md)$ is evaluated, and succeeds with an empty answer substitution. Consequently the literal $\neg offer(Z)$ is added to β at depth 1. Since no $NewTerms$ are generated, and there are no other body-decs, the algorithm terminates with $\beta = fries(Z) \text{:-} offer(Z)$. (But note that Progol4 cannot compute this hypothesis because the only head atom considered by Progol4 is $\epsilon = meal(md)$, which is compatible with no head-dec).

[12] Algorithm 3.3.1 is based on Muggleton [Mug95] Alg44 and [MF01] §4.8. For ease of presentation and comparison with future Algorithms, the standardisation of B and e has been lifted to the Cover-Set-Loop. In order to give an enhanced behaviour (closer to the actual Progol4 implementation), selection of the head-dec used to construct the MSH β_i has also been lifted to the Cover-Set Loop. Instead of selecting just one head-dec, as in earlier accounts, all possibilities are tried in turn and the best is chosen.

[13] Algorithm 3.3.3 is based on [Mug95] Alg40 and [MF01] §4.5. To give an enhanced behaviour (closer to the real Progol4), the standardisation of B and e and the head-dec selection has been lifted to the Cover-Set loop.

Begin Algorithm 3.3.3 (BOTTOMSET - [Mug95, MF01][13]).

```
Input     given $\mathcal{B}$, $\epsilon$, $m$, $M^-$
            let $InTerms = \emptyset$, $\beta = \emptyset$, and $a = schema(m)$
            assert $a\theta = \epsilon$ for some substitution $\theta$
Head        for each binding $V/t \in \theta$
                if $V$ is #type then
                    let $a = a\{V/t\}$
                else
                    let $a = a\{V/hash(t)\}$
                if $V$ is +type then
                    let $InTerms = InTerms \cup \{t\}$
            let $\beta = \beta \cup \{a\}$
Body        repeat $N_D$ times
                let $NextTerms = \emptyset$
                for each body-dec $m' \in M^-$
                    let $d = schema(m')$, $\{p_1, \ldots, p_n\} = type(m')$, and $r = recall(m')$
                    for each substitution $\theta' = \{V_1/t_1, \ldots, V_n/t_n\}$
                        where $\{V_1, \ldots, V_n\}$ is the set of +variables in $d$
                        and $t_i \in InTerms$ for all $1 \leq i \leq n$
                        repeat $r$ times
                            for each ground substitution $\theta'' \in \mathcal{S}\mathrm{LD}(\mathcal{B}, ?(d, p_1, \ldots, p_n)\theta')$
                                for each binding $V/t \in \theta'\theta''$
                                    if $V$ is #type then
                                        let $d = d\{V/t\}$
                                    else
                                        let $d = d\{V/hash(t)\}$
                                    if $V$ is -type then
                                        let $NextTerms = NextTerms \cup \{t\}$
                                let $\beta = \beta \cup \{\neg d\}$
                let $InTerms = InTerms \cup NextTerms$
Output    return $\beta$
```

End Algorithm (BOTTOMSET)

MDIE has been successfully applied to a number of real-world applications, but the strategy of considering only the trivial MSH head atom ϵ, means that the head of every hypothesised clause will contain the same predicate as the corresponding seed example. This restriction is called **Observation Predicate Learning** (OPL), and has been found to limit the applicability of ILP systems to real problems. TCIE is an extension of MDIE that addresses this limitation.

3.3.2 Theory Completion with Inverse Entailment and Progol5

TCIE is an extension of MDIE that supports the computation of MSH head atoms other than the standardised seed example ϵ. The techniques of TCIE were recently implemented in the ILP system Progol5, whose operation is shown in Algorithm 3.3.4. An additional routine called STARTSET is now added alongside BOTTOMSET to assist in the construction of the MSH. The STARTSET and BOTTOMSET routines compute respectively the head and body atoms of each MSH. As shown in Algorithm 3.3.4 below, the Cover-Set Loop is identical to that of Progol4, except that the head atom passed to BOTTOMSET is not ϵ, but an atom α_j determined by STARTSET.

It remains only to describe how the candidate head atoms are computed by the STARTSET routine. First recall from Definition 16 that the atoms in the head $Bot^+(B, e)$ of the Bottom Set are the negations of the negative ground literals deducible from B and $\overline{e}$, or equivalently from $\mathcal{B}$ and $\neg\epsilon$ The approach used by STARTSET to compute these atoms is based on the Prolog Technology Theorem Prover (PTTP) [Sti88] method of **contrapositives**; which is essentially a technique for propagating negative information backwards through the program clauses. The concept is best illustrated by example.

Take the clause a :- b. Although this clause is conventionally used to conclude a on the strength of b, by analogy with the classical equivalence $a \leftarrow b$ iff $\neg b \leftarrow \neg a$, it can equally be used to conclude $\neg b$ on the strength of $\neg a$. In the PTTP context this inference is achieved by augmenting the original program with clauses containing *new* predicates (p^*/n) to represent the negations of existing predicates p/n. For instance, the clause a :- b would yield the contrapositive b^* :- a^*.

Generalising, every n atom definite clause yields $n - 1$ contrapositive variants, each of which is obtained by exchanging a different body atom with the head. For example, by analogy with the classical equivalences $a \leftarrow b \wedge c \equiv \neg b \leftarrow \neg a \wedge c \equiv \neg c \leftarrow \neg a \wedge b$, the clause a :- b, c yields the contrapositives b^* :- a^*, c and c^* :- a^*, b.

Example 3.3.5. The query $?fries^*(rz)$ succeeds from the clause $meal(X)$:- $fries(X), burger(X)$ together with its contrapositive $fries^*(X)$:- $meal^*(X)$, $burger(X)$ and the facts $burger(rz)$ and $meal^*(rz)$, simulating the inference that if one had a *burger* at *theRitz* but not a *meal*, then one could not have had *fries*.

In order to compute the negative unit consequences of $\mathcal{B}$ and $\neg\epsilon$, one approach is to form contrapositives for every rule in the definite theory $\mathcal{B}$ and for the negative ground fact $\neg\epsilon$. As shown in Definition 27 below, this augmented theory will be called the *contrapositive*

[14]Algorithm 3.3.4 is based on Muggleton [MB00] §2.3. For ease of presentation and comparison with future Algorithms, the standardisation of B and e has been lifted to the Cover-Set Loop, thereby avoiding the need to perform Skolemisation twice and dispensing with the unnecessary bridging clause. The selection of head-decs has been similarly lifted in order to avoid an unnecessary restriction in the original description and to give an enhanced beaviour closer to the actual Progol5 implementation.

Begin Algorithm 3.3.4 (Progol5 - [MB00][14]).

```
Input              given $B$, $E^+$, $E^-$, $M$
check consist.       assert $Consistent(B \cup E^+ \cup E^-)$
remove cover         let $E^+ = E^+ - Cover(B, E^+)$
Cover-Set Loop       while $E^+ \neq \emptyset$
select seed              choose seed example $e \in E^+$
standardise              let $\langle \mathcal{B}, \epsilon \rangle = Standardise(B, e)$
select head-dec          for each head-dec $m_i \in M^+$ such that $m_i \succeq \epsilon$
contrapos.                   let $\mathcal{A}_i = \mathcal{S}\text{TART}\mathcal{S}\text{ET}(\mathcal{B}, \epsilon, m_i)$
select ead atom              for each atom $\alpha_j \in \mathcal{A}_i$
bottom set                       let $\beta_{ij} = \mathcal{B}\text{OTTOM}\mathcal{S}\text{ET}(\mathcal{B}, \alpha_j, m_i, M^-)$
search                           let $h_{ij} = \mathcal{S}\text{EARCH}(\beta_{ij}, B, E^+, E^-, M)$
assert best hyp.         add to $B$ the $h_{ij}$ with greatest $Compression$
remove cover             let $E^+ = E^+ - Cover(B, E^+)$
Output             return $B$
```

End Algorithm (Progol5)

expansion of $\mathcal{B}$ and $\neg\epsilon$, and denoted $\mathcal{B}^*_\epsilon$. The negative unit consequences are determined by running atomic starred queries of the form $p^*(\dots)$ against the theory $\mathcal{B}^*_\epsilon$. Each successful answer substitution θ identifies the atom $\neg p(\dots)\theta$ as a consequence of $B \cup \overline{e}$, and finally, taking the negation gives $p(\dots)\theta \in Bot^+(B, e)$. (n.b. see Proposition 3.3.13 below).

Definition 27 (Contrapositive Expansion). Let $\mathcal{B}$ and ϵ be the result of standardising a Horn theory B and a Horn clause e. Then the *contrapositive expansion of B and e*, denoted $\mathcal{B}^*_\epsilon$, is the definite theory

$$\mathcal{B}^*_\epsilon = \mathcal{B} \cup \{\epsilon^*\} \cup \{b^*_j \text{:-}\, b^*_0, \dots, b_{j-1}, b_{j+1}, \dots, b_m \mid b_0 \text{:-}\, b_1, \dots, b_j, \dots, b_m \in \mathcal{B}\} \qquad \blacklozenge$$

Example 3.3.6. Referring to the Fast Food example shown in Figure 3.1, if B is the knowledge base K_B of Figure 2.1, and if e is the atom $meal(md)$, then $Standardise(B, e) = \langle \mathcal{B}, \epsilon \rangle = \langle B, e \rangle$, and the contrapositive expansion of B and e is

$$\mathcal{B}^*_\epsilon = K_B \cup \left\{ \begin{array}{l} meal^*(md) \\ burger^*(X) \text{:-}\, meal^*(X),\ fries(X) \\ fries^*(X) \text{:-}\, meal^*(X),\ burger(X) \\ fries^*(X) \text{:-}\, burger^*(X),\ offer(X) \\ offer^*(X) \text{:-}\, burger^*(X),\ fries(X) \end{array} \right\}$$

In practice, two restrictions are imposed on computed atoms: they must be ground and they must conform to the specified language bias. For each head-dec m, the query formed by STARTSET consists of the starred form of the schema of m, to which are appended additional type calls for each variable in the schema. These precautions ensure that each successful substitution is ground and is compatible with m. An unnecessary proliferation of contrapositives is avoided by observing that contrapositives need only be formed for rules whose head predicate occurs on a path in the call-graph of $\mathcal{B}$ that starts on the predicate of ϵ and ends on the predicate in the schema of m.

```
Begin Algorithm 3.3.7 (STARTSET - [MB00][15]).
  Input        given $\mathcal{B}$, $\epsilon$, $m$
                 let $\mathcal{A} = \emptyset$
  Contrapos.     let $\mathcal{B}^*_\epsilon = \mathcal{B} \cup \{\epsilon^*\} \cup \{b_j^* \text{:-} b_0^*, \ldots, b_{j-1}, b_{j+1}, \ldots, b_n \mid b_0 \text{:-} b_1, \ldots, b_j, \ldots, b_n \in \mathcal{B}\}$
                   where $n, j > 0$ and $pred(b_0)$ occurs on some path in the call-graph of $\mathcal{B}$,
                   starting from $pred(\epsilon)$ and ending on $pred(m)$
                 let $a = schema(m)$ and $\{t_1, \ldots, t_p\} = type(m)$
  Query          for each grounding substitution $\theta \in S\text{LD}(\mathcal{B}^*_\epsilon, ?a^*, t_1, \ldots, t_p)$
                     let $\mathcal{A} = \mathcal{A} \cup \{a\theta\}$
  Output       return $\mathcal{A}$
End Algorithm (STARTSET)
```

Thus given a definite theory $\mathcal{B}$, a ground atom ϵ, and a head-dec m, the operation of the STARTSET routine is described in Algorithm 3.3.7 below. The set $\mathcal{A}$ denotes the initially empty set of candidate head atoms that will be returned by the routine. First the required contrapositives are constructed from $\mathcal{B}$ and ϵ, and then the necessary query is formed from the schema and type information in m. The atoms in $\mathcal{A}$ are determined by removing the star from each successful instance of the query.

Example 3.3.8 (Failure of STARTSET). The STARTSET computation resulting from Fast-Food example of Figure 3.1 with the seed example $meal(md)$, is shown below. This query is constructed from the only available head-dec modeh$[fries(+bistro)]$.

$$\begin{array}{ll}
\multicolumn{2}{c}{?\,fries^*(Z), bistro(Z)} \\
?\,meal^*(Z), burger(Z), bistro(Z) & ?\,burger^*(Z), offer(Z), bistro(Z) \\
?\,burger(md), bistro(md) & ?\,meal^*(Z), fries(Z), offer(Z), bistro(Z) \\
?\,fries(md), offer(md), bistro(md) & ?\,fries(md), offer(md), bistro(md) \\
?\,\blacksquare & ?\,\blacksquare
\end{array}$$

Because both branches of the computation fail, no candidate head atoms are returned by STARTSET, and no hypothesis is computed by Progol5.

Example 3.3.9 (Success of STARTSET). If the additional fact $burger(md)$ had been added to the knowledge base K_B, then the computation depicted in Example 3.3.8 above, would have produced the additional branch shown below, leading to the successful answer substitution $\{Z/md\}$

$$\begin{array}{l}
?\,fries^*(Z), bistro(Z) \\
?\,meal^*(Z), burger(Z), bistro(Z) \\
?\,burger(md), bistro(md) \\
?\,bistro(md) \\
?\,\square
\end{array}$$

Because this branch of the computation succeeds, the candidate head atom $fries(md)$ is returned by STARTSET, and the hypothesis $H = \{fries(X) \text{:-} offer(X)\}$ is computed by Progol5.

[15] Algorithm 3.3.7 is based on Muggleton [MB00] §2.1. In this paper the notation $p^*(\ldots)$ is used instead of $non\_p(\ldots)$.

3.3.3 Incompleteness of StartSet

A surprising fact revealed by Example 3.3.8 is that the STARTSET routine is fundamentally incomplete with respect to its intended purpose of computing atoms in the head of the Bottom Set, as formalised in Proposition 3.3.10 below. Moreover, because the the atom α is not vacuous, it is also contained in the refined Bottom Set, and therefore STARTSET is also incomplete with respect to refined Bottom Generalisation.

Proposition 3.3.10 (Incompleteness of StartSet w.r.t. Bottom Generalisation). *For some Horn theory B and some Horn clause e there exists an atom $\alpha \in Bot^{+}(B,e)$ that cannot be computed by STARTSET.*

Proof. Let $B = K_B$, let $e = meal(md)$, and let $\alpha = fries(md)$. Proof is immediate from Example 3.2.3, which shows that $\alpha \in Bot^{+}(B,e)$, and from Example 3.3.8, which shows that the STARTSET algorithm terminates without computing α – even though α lies within the available language and search bias. ∎

The incompleteness identified above is related to the notion of c-derivation, which was seen earlier to characterise the hypotheses derivable by Bottom Generalisation. Referring to Figure 3.1, taking $B = K_B$ and $e = meal(md)$ and $h = fries(X)\,\text{:-}\,offer(X)$, it was shown in Example 3.2.3 that h is derivable by Bottom Generalisation from B and e. By the results given previously, this is sufficient to ensure the existence of a c-refutation $(B \cup \overline{e}, h) \vdash_c \Box$.

It is instructive to construct a c-refutation for the example in Proposition 3.3.10, and one such refutation is shown in Figure 3.2 below. In this diagram input clauses 1 and 2 resolve to give resolvent clause 3. Resolvent clause 3 then resolves with input clause 4 to give resolvent clause 5, and so on until the empty clause is derived.

```
1. :- meal(md)
   |
   |— — —  2. meal(X) :- fries(X), burger(X)
   |
3. :- fries(md), burger(md)
   |
   |— — —  4. burger(Y) :- fries(Y), offer(Y)
   |
5. :- fries(md), offer(md)
   |
   |— — —  6. fries(X) :- offer(X)
   |
7. :- offer(md)
   |
   |— — —  8. offer(md)
   |
   □
```

Figure 3.2: C-Derivation

It is important to note, however, that the definition of a clause used by Plotkin is that of a *set* of literals; and therefore the merging of identical literals must be enforced in each resolution step. It is interesting that in this example every refutation in which the hypothesis h is used only once, requires at least one merge. In the c-derivation shown in Figure 3.2 above, a merge of two $fries(md)$ atoms occurs in the resolvent clause 3, and so the hypothesis is only needed once: i.e. clause 6.

If a c-refutation in which there is no merging of literals is called a *c*-refutation*, then it remains to show that an atom a is computed by STARTSET only if there exists a c*-refutation of $B \cup \overline{e}$ with respect to a (denoted $(B \cup \overline{e}, a) \vdash_c^* \Box$), and, moreover, that a hypothesis h is computed by Progol5 only if $(B \cup \overline{e}, h) \vdash_c^* \Box$. But note, however, that the converse of this conjecture is false because many of the vacuous head atoms not computed by STARTSET have c*-derivations. For example, given $B = \{ \text{:-} b, c\}$ and $e = a \text{:-} b$, then the hypothesis $h = c$ is derivable by Bottom Generalisation but is *not* computed by Progol5 even though there is a c*-refutation (b from $\overline{e}$ resolves with $\text{:-} b, c$ from B to give $\text{:-} c$, which resolves with c from h to give $\Box$).

3.3.4 Soundness of StartSet

Proposition 3.3.13 below demonstrates the soundness of the STARTSET procedure with respect to Refined Bottom Generalisation, using Lemmas 3.3.11 and 3.3.12, both of which are shown by induction of the length of successful computations.

Lemma 3.3.11 (Queries with no starred calls). *Let $\mathcal{B}$ and ϵ be the result of standardising a Horn theory B and a Horn clause e, and let $\mathcal{B}_\epsilon^*$ denote the contrapositive expansion. Let $Q = ?c_1, \ldots, c_n$ be any query containing no starred calls. Then*

$$\mathcal{B}_\epsilon^* \vdash Q\sigma \textit{ implies } \mathcal{B} \vdash Q\sigma \textit{ with an } \text{identical } \textit{refutation}$$

Proof. By induction on the length of the refutation. Let the query $?c_1, \ldots, c_n$ where $n \geq 1$ succeed from $\mathcal{B}_\epsilon^*$ with a refutation R of length p and substitution σ. [**Base Case**]. If $p = 1$ then $n = 1$ and the query $?c_1$ succeeds from $\mathcal{B}_\epsilon^*$ in one step by resolving with some fact F under the unifier σ. Since F must be non-starred in order to unify with c_1, from the definition of $\mathcal{B}_\epsilon^*$ it must be that F is already contained in $\mathcal{B}$ (as the only fact added to $\mathcal{B}$ is starred: i.e. ϵ^*). Thus the query succeeds identically from $\mathcal{B}$. [**Induction Step**]. If $p = k$ then the first step of the refutation consists of the reduction of the first call c_1 of the query $?c_1, \ldots, c_n$ with some clause C in $\mathcal{B}_\epsilon^*$ of the form $c \text{:-} d_1, \ldots, d_m$, to yield the derived query $?d_1\sigma', \ldots, d_m\sigma', c_2\sigma', \ldots, c_n\sigma'$, where σ' is the unifier of c_1 and c. Since the head of C must be non-starred in order to unify with c_1, from the definition of $\mathcal{B}_\epsilon^*$ it must be that C is already contained in $\mathcal{B}$ (as all of the clauses added to $\mathcal{B}$ have a starred atom in the head). Note also from the definition of $\mathcal{B}_\epsilon^*$ that every d_i is non-starred (as starred predicates are fresh predicates which by definition do not appear in $\mathcal{B}$). The remaining steps in the refutation then consist of the reduction of the derived query $?d_1\sigma', \ldots, d_m\sigma', c_2\sigma', \ldots, c_n\sigma'$, which is known to succeed from $\mathcal{B}_\epsilon^*$ in $k-1$ steps with substitution σ'' such that $\sigma'\sigma'' = \sigma$. Since this query contains no starred atoms, by the induction hypothesis it has an identical refutation from $\mathcal{B}$. Thus the original query succeeds with an identical refutation R from $\mathcal{B}$. This completes the proof by induction. ∎

Lemma 3.3.12 (Queries with one starred call). *Let $\mathcal{B}$ and ϵ be the result of standardising a Horn theory B and a Horn clause e, and let $\mathcal{B}^*_\epsilon$ denote the contrapositive expansion. Let $Q = ?b^*, c_1, \ldots, c_n$ be any query containing exactly one starred call b^*. Then*

$$\mathcal{B}^*_\epsilon \vdash Q\sigma \text{ implies } \mathcal{B} \cup \{b\sigma\} \models \epsilon$$

Proof. By induction on the length of the refutation. Let the query $?b^*, c_1, \ldots, c_n$ where $n \geq 0$ succeed from $\mathcal{B}^*_\epsilon$ with a refutation R of length p and substitution σ. [**Base Case**]. If $p = 1$ then $n = 0$ and the query $?b^*$ succeeds in one step by resolving with the only starred fact ϵ^* under the unifier σ. Since ϵ^* is ground $b^*\sigma \doteq \epsilon^*$ so removing stars $b\sigma = \epsilon$ so by reflexivity $b\sigma \models \epsilon$ and by monotonicity $\mathcal{B} \cup \{b\sigma\} \models \epsilon$. [**Induction Step**]. If $p = k$ then either: i) the first call b^* succeeds, as in the base case, by resolving with ϵ^* under the unifier σ' and the derived query $?c_1\sigma', \ldots, c_n\sigma'$ succeeds in $k-1$ steps with substitution σ'' such that $\sigma'\sigma'' = \sigma$. Since ϵ^* is ground $\epsilon^* \doteq b^*\sigma' \doteq b^*\sigma'\sigma'' \doteq b^*\sigma$ and thus $\mathcal{B} \cup \{b\sigma\} \models \epsilon$ as in the base case. Or else ii) the call b^* resolves with a clause C^* in $\mathcal{B}^*_\epsilon$ of the form $b^* \text{:-} d^*, e_1, \ldots, e_m$ under the unifier σ', and the derived query $?d^*\sigma', e_1\sigma', \ldots, e_m\sigma', c_1\sigma', \ldots, c_n\sigma'$ succeeds in $k-1$ steps with substitution σ'' such that $\sigma'\sigma'' = \sigma$. Now this case has three implications: By the inductive hypothesis $\mathcal{B} \cup \{(d\sigma')\sigma''\} \models \epsilon$ or equivalently $B \cup \{d\sigma\} \models \epsilon$ so by the entailment theorem $\mathcal{B} \models \forall d\sigma \rightarrow \epsilon$ [1]. From the definition of $\mathcal{B}^*_\epsilon$, corresponding to C^* there is a clause C in $\mathcal{B}$ of the form $d \text{:-} b, e_1, \ldots, e_m$ so that $\mathcal{B} \models \forall(e_1 \wedge \ldots \wedge e_m \wedge b \rightarrow d)$ and by substitution $\mathcal{B} \models \forall(e_1\sigma \wedge \ldots \wedge e_m\sigma \wedge b\sigma \rightarrow d\sigma)$ [2]. By lemma 3.3.11 the non-starred calls $e_1\sigma', \ldots, e_m\sigma'$ succeed also from $\mathcal{B}$ with substitution σ'', so by the soundness of SLD resolution $\mathcal{B} \models \forall e_j\sigma'\sigma'' \equiv \forall e_j\sigma$ and so $\mathcal{B} \models \forall(e_1\sigma \wedge \ldots \wedge e_n\sigma)$ [3]. Therefore $\mathcal{B} \models \forall(b\sigma \rightarrow d\sigma)$ from [2] and [3] and consequently $\mathcal{B} \models \forall b\sigma \rightarrow \epsilon$ from [1]. And thus $\mathcal{B} \cup \{b\sigma\} \models \epsilon$ by the entailment theorem. This completes the proof by induction.

▮

Proposition 3.3.13 (Soundness of StartSet w.r.t. Refined Bottom Generalisation). *Given a Horn theory B, a Horn clause e, and a head-dec m, let $\langle\mathcal{B}, \epsilon\rangle = Standardise(B, e)$, and let α be any atom other than ff, then*

$$\alpha \in STARTSET(\mathcal{B}, \epsilon, m) \text{ implies } \alpha \in Bot^*(B, e)$$

Proof. First let $\mathcal{B}^*_\epsilon$ denote the contrapositive extension of $\mathcal{B}$ and ϵ, let $a = schema(m)$ denote the schema of m, and let $\{t_1, \ldots, t_p\} = type(m)$ denote the types of the variables in a, and define the query $Q = ?a^*, t_1, \ldots, t_p$. Now, if $\alpha \in \mathcal{S}\text{TART}\mathcal{S}\text{ET}(\mathcal{B}, \epsilon, m)$ then by Algorithm 3.3.7 it follows $\mathcal{B}^*_\epsilon \vdash Q\sigma$ for some grounding substitution σ such that $a\sigma \doteq \alpha$. Therefore $\mathcal{B} \cup \{\alpha\} \vdash ?\epsilon$ by Lemma 3.3.12. Consequently $\alpha \in Bot^*(B, e)$ by Definition 25. ▮

3.3.5 On the Computation of Vacuous Literals

Many vacuous literals are not computed by STARTSET. For example, if $B = \{\text{:-} c\}$ and $e = a$, then the literal c is in $Bot^+(B, e)$, but is not computed by STARTSET. In this case $\mathcal{B}^*_\epsilon$ consists of the three clauses $\texttt{ff} \text{:-} c$ and $c^* \text{:-} \texttt{ff}^*$ and a^*. Because the atom $\texttt{ff}^*$ is not present, the query $?c^*$ fails. In fact, the atom ff is only ever present when the example is a negative clause; in which case $\epsilon = \texttt{ff}$ by definition 22 and so $\epsilon^* = \texttt{ff}^*$ is included in $\mathcal{B}^*_\epsilon$ by Definition 27. If it is really desired to compute vacuous literals, then one of two simple methods, described below, may be used. Each involves replacing the contrapositive expansion with an alternative construction called respectively the *Strong* and *Weak Contrapositve Extension* - which turn out to be equivalent.

a) Strong Contrapositve Extension

The strong contrapositve extension is obtained by adding the starred atom ff^* to the contrapositive expansion, as shown in Definition 28 below.

Definition 28 (Strong Contrapositive Extension). Given a Horn theory B and a Horn Clause e, the *(strong) contrapositive extension*, denoted B_e^+, is the Horn theory:

$$B_e^+ = \mathcal{B}_\epsilon^* \cup \mathsf{ff}^* \qquad \blacklozenge$$

If B_e^+ is used in STARTSET in place of $\mathcal{B}_\epsilon^*$ then the contrapositives of integrity constraints will be able to participate in *any* derivation. In this case STARTSET will be sound with respect to the computation of atoms in the head of the Bottom Set, as shown by making the following changes to the results above:

1. In Lemma 3.3.11 the bracket in the base case becomes: "(the only facts added to $\mathcal{B}$ are starred: i.e. ϵ^* or ff^*)".
2. In Lemma 3.3.12 the result is changed to: $B_e^+ \vdash Q\sigma$ implies $\mathcal{B} \cup \{b\sigma\} \models \epsilon$ or $\mathcal{B} \cup \{b\sigma\} \models \mathsf{ff}$; and the base case includes the extra case: "...or by resolving with the starred fact ff^*, so that $b = \mathsf{ff}$, hence $b \models \mathsf{ff}$ and so $\mathcal{B} \cup \{b\sigma\} \models \mathsf{ff}$ using monotonicity and the fact b is already ground".
3. In Proposition 3.3.13 the result is changed to: $\alpha \in \mathcal{S}\text{TART}\mathcal{S}\text{ET}(\mathcal{B}, \epsilon, m)$ implies $\alpha \in Bot(B, e)$; and the last line of the proof is changed to: "Therefore $\mathcal{B} \cup \{\alpha\} \vdash ?\epsilon$ or $\mathcal{B} \cup \{\alpha\} \vdash ?\mathsf{ff}$ by Lemma 3.3.12. Consequently $\alpha \in Bot(B, e)$ by Lemma 3.2.14".

b) Weak Contrapositve Extension

The weak contrapositve extension avoids the need for explicit standardisation. Formally, this approach can be built upon on the notion of *complementary extension*, and by extending the notion of contrapositive variants to Horn clauses and Horn theories. These notions are formalised in Definitions 29, 30, 31, and 32 below.

Definition 29 (Weak Contrapositive Extension). Given a Horn theory B and a Horn Clause e, the *(weak) contrapositive extension*, denoted B_e^-, is the Horn theory:

$$B_e^- = B_e \cup Contra(B_e) \qquad \blacklozenge$$

Definition 30 (Complementary Extension). Given a Horn theory B and a Horn Clause e, the *complementary extension* B_e is the Horn theory defined as follows:

$$B_e = B \cup \overline{e} \qquad \blacklozenge$$

Definition 31 (Contrapositive Variants of a Horn theory). Let $T = \{T_1, \ldots, T_n\}$ be a Horn theory. Then the *contrapositive variants* of T, written $Contra(T)$, are defined as the definite theory:

$$Contra(T) = Contra(T_1) \cup \ldots \cup Contra(T_n) \qquad \blacklozenge$$

Definition 32 (Contrapositive Variants of a Horn clause). Let C be a Horn clause (containing no starred predicates). Then the *contrapositive variants* of C, written $Contra(C)$, are defined as the definite theory containing all clauses of the form:

$$C - \{C_0, \neg C_j\} \cup \{C_j^*, \neg C_0^*\} \quad \text{for some } 1 \leq j \leq n, \quad \text{for definite } C = \{C_0, \neg C_1, \ldots, \neg C_n\}$$

or

$$C - \{\neg C_j\} \cup \{C_j^*\} \quad \text{for some } 1 \leq j \leq n, \quad \text{for negative } C = \{\neg C_1, \ldots, \neg C_n\}$$

♦

It can be shown that the two approaches described above are equivalent in the sense that they have (almost) identical success sets under SLD resolution. The idea is illustrated in Example 3.3.14 below, in which the background knowledge B states that: i) every one who is *liked* by *boss* receives a pay *bonus*, and ii) no one can be *lazy* and *keen* at the same time. And the example e is challenging the learner to explain why every *keen* person receives a pay *bonus*. Let k_x denote the Skolem constant introduced for the variable X. Note that for ease of comparison some clauses have been labelled with lower case letters.

Example 3.3.14 (Equivalence of Strong and Weak Approaches).

$$B = \left\{ \begin{array}{l} bonus(X) \,\text{:-}\, likes(boss, X) \\ \text{:-}\, keen(X), lazy(X) \end{array} \right\} \qquad e = bonus(X) \,\text{:-}\, keen(X)$$

$$B_e^- = \left\{ \begin{array}{ll} a. & bonus(X) \,\text{:-}\, likes(boss, X) \\ b_1. & \text{:-}\, keen(X), lazy(X) \\ b_2. & \text{:-}\, bonus(k_x) \\ c. & keen(k_x) \end{array} \right\} \cup \left\{ \begin{array}{ll} d. & likes^*(boss, X) \,\text{:-}\, bonus^*(X) \\ e. & keen^*(X) \,\text{:-}\, lazy(X) \\ f. & lazy^*(X) \,\text{:-}\, keen(X) \\ g. & bonus^*(k_x) \end{array} \right\}$$

$$\mathcal{B} = \left\{ \begin{array}{l} bonus(X) \,\text{:-}\, likes(boss, X) \\ \mathsf{ff} \,\text{:-}\, keen(X), lazy(X) \\ keen(k_x) \end{array} \right\} \qquad \epsilon = bonus(k_x)$$

$$B_e^+ = \left\{ \begin{array}{ll} a'. & bonus(X) \,\text{:-}\, likes(boss, X) \\ b'. & \mathsf{ff} \,\text{:-}\, keen(X), lazy(X) \\ c'. & keen(k_x) \\ h. & \mathsf{ff}^* \end{array} \right\} \cup \left\{ \begin{array}{ll} d'. & likes^*(boss, X) \,\text{:-}\, bonus^*(X) \\ e'. & keen(X)^* \,\text{:-}\, \mathsf{ff}^*, lazy(X) \\ f'. & lazy(X)^* \,\text{:-}\, \mathsf{ff}^*, keen(X) \\ g'. & bonus^*(k_x) \end{array} \right\}$$

Clauses a, c, d, g are identical a', c', d', g' respectively. Clauses b_1, b_2, b' are all redundant: the former can never participate in an SLD derivation because they are negative, and the latter can never participate because by definition the atom ff does not appear in the body of any clause. After unfolding with the atom ff^*, clauses e', f' are identical to e, f. The only difference concerns the atom ff^*, which always succeeds from B_e^+ (which always contains the fact ff^*) but never succeeds from B_e^- (which never even mentions the symbol ff), but in practice this is inconsequential.

Chapter 4

Abductive Logic Programming and ASLD

In this chapter task of ALP [KKT92] is defined and the Abductive SLD (ASLD) procedure of Kakas and Mancarella [KM90] is described. The purpose of this chapter is to recall the task of ALP and to provide a brief overview of the ASLD procedure. A detailed analysis is not required for the purposes of this report, and the references cited above provide a good introduction to the concepts required. The presentation of ALP below, is analogous to the presentation of ILP in the previous section. The main difference is that ALP considers normal rather than definite logic programs, and the semantics of ALP is therefore referred to a canonical model construction. But for the applications considered in this report, the choice of semantics will not play an important role.

Abduction is concerned with the explanation of goals with respect to a prior theory. The inputs to the abductive task are a background *theory* T, a *goal* g to be proved, a set of *integrity constraints* IC that must be satisfied, and a set of *abducible predicates* A from which to construct hypotheses. These inputs will collectively be called an *abductive context*, as formalised in definition 33 below. The output of the abductive task is a *hypothesis* Δ that is a conjunction of ground abducible atoms which when added to T 'explain' g and 'satisfy' IC. The notion of abductive explanation is formalised in definition 34 below, with respect to a chosen canonical model construction $\mathcal{M}$.

Definition 33 (Abductive Context). An *abductive context* is a 4-tuple $\langle T, g, IC, A\rangle$ where T is a normal logic program, g is a normal query, IC set of normal denials, and A is a set of predicate symbols. ♦

Definition 34 (Abductive Explanation - [KD02][1]). Given an abductive context $\langle T, g, IC, A\rangle$ and a canonical model construction $\mathcal{M}$. Let $\mathcal{A}$ denote the set of ground instances of predicates in A. An *abductive explanation* is a conjunction $\Delta \subseteq \mathcal{A}$ such that: for some substitution θ the following three conditions hold:

i. $T \cup \Delta \models_{\mathcal{M}} g\theta$ **- explanation**

ii. $T \cup \Delta \models_{\mathcal{M}} IC$ **- integrity**

iii. "$T \cup \Delta \not\models_{\mathcal{M}} \Box$" **- satisfiability** ♦

[1]Definition 34 is based on [KD02] Defn 3.1.

Theory	Goal	Abducibles
$T = K_B$	$g = \{meal(md)\}$	$A = \{fries\}$

Integrity Constraints	
$IC = \left\{ \begin{array}{l} \text{:-}\, fries(V),\ \sim bistro(V) \\ \text{:-}\, meal(rz),\ fries(W) \end{array} \right\}$	ic1 ic2

$\Rrightarrow$

Explanation
$\Delta = \{fries(md)\}$

Figure 4.1: Fast-Food Example: an ALP Problem

Briefly Δ is an abductive explanation if under some substitution g is true in the cannonical model (in some chosen semantics) of T and Δ, and the integrity constraints are also true in this model (a view of integrity sometimes referred as "epsitemic" [KKT92]). The last condition is not intended to be taken literally, but is there merely for emphasisis and should be read as saying "the canonical model exists" (which is implicit from the first two conditions).

Should the inputs admit of more than one abductive explanation, additional preference criteria are applied to eliminate the least favourable. Common desiderata dictate that the explanation should be no larger than necessary (minimal) and as fundamental as possible (basic) - as formalised in definitions 35 and 36 below.

Definition 35 (Minimal). An explanation Δ of $\langle T, g, IC, \mathcal{A}\rangle$ is **minimal** iff for no explanation Δ' of $\langle T, g, IC, \mathcal{A}\rangle$ is it the case that $\Delta' \subset \Delta$. An explanation Δ of $\langle T, g, IC, \mathcal{A}\rangle$ is **locally minimal** iff it is a minimal explanation of $\langle T', g, IC, \mathcal{A}\rangle$ for some $T' \subseteq T$. ⧫

Definition 36 (Basic). An explanation Δ of $\langle T, g, IC, \mathcal{A}\rangle$ is **basic** iff for every explanation Δ' of $\langle T, \Delta, IC, \mathcal{A}\rangle$ it is the case that $\Delta \subseteq \Delta'$. ⧫

Informally an explanation is minimal if it contains no redundant atoms, and locally minimal if it is minimal with respect to some subset of the theory (i.e. omitting redundancy explicit in the theory itself). An explanation is basic if it can itself be otherwise explained only by adding redundant atoms (i.e. in terms of non-minimal explanations) - see [CP86] for a more general account.

Example 4.0.15. An illustrative example is given in figure 4.1 above, where T is the knowledge base K_B from figure 2.1. Hypothesis Δ is a valid explanation by definition 34 because it consists of ground instances of abducible predicates in $\mathcal{A}$, and together with T it satisfies IC and entails g.

In addition, $\Delta = \{fries(md)\}$ is minimal and basic according to definitions 35 and 36. Note that while $\Delta' = \{fries(md), fries(bk)\}$ is an alternative explanation, Δ' is not minimal because $\Delta \subset \Delta'$. Note also Δ is basic even though it is (trivially) explained by Δ'.

An important approach in the implementation of ALP is that of generalising traditional SLD methods to support the construction of abductive hypotheses. A successful example of this approach is the ASLD proof procedure of Kakas and Mancarella [KM90], which iteratively interleaves computations consisting of an **abductive phase** in which abducibles are hypothesised, and a **consistency phase** in which hypotheses are incrementally tested against integrity constraints.

ASLD takes as input an abductive context $\langle T, g, IC, A\rangle$ in which (without loss of generality) no abducible predicate (in A) is defined (in T). If necessary this can be ensured by the following transformation stated in [KKT92]: If any abducible predicate $p/n \in \mathcal{A}$ is defined

$\{\}?\ meal(md)$
$\{\}?\ fries(md), burger(md)$
$\{fries(md), fries^*(rz)\}?\ burger(md)$
$\{fries(md), fries^*(rz)\}?\ fries(md), offer(md)$
$\{fries(md), fries^*(rz)\}?\ offer(md)$
$\{fries(md)\}?\ \square$

$\{fries(md)\}?\ meal(rz)$
$\{fries(md)\}?\ fries(rz), burger(rz)$
$\{fries(md), fries^*(rz)\}?\ \blacksquare$

$\{fries(md), fries^*(rz)\}?\ \backsim bistro(rz)$
$\{fries(md), fries^*(rz)\}?\ \blacksquare$

Figure 4.2: ASLD: Burger Example (Abductive and Consistency Phases)

in T, then p can be replaced everywhere except in $\mathcal{A}$ by the new predicate p^δ/n and the new clause $p^\delta(X_1,\ldots,X_n)$:- $p(X_1,\ldots,X_n)$ added to T. In addition, every integrity constraint is assumed to mention at least one abducible predicate.

The main abductive computation is initiated by running query g on program T, and is identical to standard SLD resolution *except* when an abducible a is selected. Whereas conventional SLD would be forced to fail the current computation and backtrack, ASLD treats this abducible as a candidate hypothesis, and invokes a consistency computation, aiming to show that the addition of this abducible to the current hypothesis Δ, would not violate integrity.

The consistency computation is designed so as to avoid the need to test *all* integrity constraints against each abducible, and consists of a sequence of checks, one for each integrity constraint that resolves with the abducible under investigation. Every such resolvent is run as a query, each of which must finitely fail for the integrity check to succeed. If necessary this failure can be expedited by a commitment *not* to abduce some other atom b, and this eventuality is recorded by adding b^* to Δ, indicating that all subsequent calls to b should immediately fail.

If all consistency checks are passed (i.e. they all end in failure), then the abductive computation continues with the abducible a added to Δ, indicating that all subsequent calls to a should immediately succeed. If any consistency check is not passed (i.e. it ends in success) then the current computation is failed and the backtracking mechanism is invoked.

There is an obvious similarity between consistency mechanism described above, and the familiar NAF mechanism as used by SLDNF. Indeed, in the general ASLD procedure, NAF is simply treated as a special case of abduction[2]. The correct handling of negation in this way requires that subordinate abductive computations be systematically invoked by consistency computations, leading to arbitrary nestings.

Example 4.0.16. The ASLD computation resulting from figure 4.1 is shown in figure 4.2, where each query is preceded by the set of currently abduced atoms. The abductive computation, shown in the single LHS box is initiated by the goal query $Q = ?meal(md)$, with an empty set of abduced atoms. The subsequent computation is identical to that of a Prolog interpreter until the abducible $fries(md)$ is selected in the third line.

At this point the consistency computation shown in the double RHS box is invoked. Each half begins with the query obtained from the resolvent of $fries(md)$ with an integrity constraint. The topmost check is run on T under the assumption $fries(md)$ and fails with

[2]More specifically, for each predicate p/n a new predicate p^*/n is added to $\mathcal{A}$ to represent the negation of p, and the canonical integrity constraints :-$p(X_1,\ldots,X_n), p^*(X_1,\ldots,X_n)$ and (implicitly) $p(X_1,\ldots,X_n) \vee p^*(X_1,\ldots,X_n)$ are added to IC. Any NAF literals of the form $\backsim p(t_1,\ldots,t_n)$ are then replaced by $p^*(t_1,\ldots,t_n)$.

a commitment *not* to abduce $fries(rz)$, which is recorded by appending $fries^*(rz)$ to the hypothesis before passing it on to the next check. The lower test fails under the standard SLDNF mechanism.

Thus $fries(md)$ passes the integrity check and so the abductive computation continues with this fact having been added to the set of currently abduced atoms, along with the additional commitment to $fries^*(rz)$. The next call succeeds by standard SLD and the last by virtue of $fries(md)$ having been previously abduced. Finally, the set of positive abducibles is returned as computed abductive explanation $A = \{fries(md)\}$. In this case no further solutions are found upon backtracking.

It can be shown that (under certain restrictions), atoms contained in a Bottom Set $Bot(B, e)$ can be computed by ASLD from a theory T consisting of the Standardised background knowledge $\mathcal{B}$, with a goal consisting of the standardised example ϵ. Note that in Example 4.0.16 the ASLD procedure is shown to compute the very atom that STARTSET failed to compute in Example 3.3.8. This observation will be elaborated and further developed in the next section. (i.e. rather than expound at this stage the necessary conditions required for the method to be complete, this observation will instead be used to motivate a semantic analysis and generalisation of the Bottom based approach that will subsequently be refined in such a way as to exploit the observations of previous chapters).

Chapter 5

Hybrid Abductive Inductive Learning

This chapter introduces a novel learning approach, called Hybrid Abductive Inductive Learning (HAIL), which builds upon the previous two chapters. The purpose of this Section is to motivate the abductive, deductive and inductive components of the HAIL learning cycle, and to present an instance of this cycle that combines and generalises existing ASLD and Progol5 methods. Subsection 5.1 motivates and formalises the semantics of Kernel Set Subsumption, showing it to be a sound method of inductive inference which extends that of Bottom Generalisation. Subsection 5.2 introduces a generic proof procedure, a concrete refinement of which is provided in Subsection 5.3. This procedure is illustrated in Subsection 5.3.3 by means of two worked examples. Finally, Subsection 5.3.4 proposes a simple extension of the procedure.

5.1 The Semantics of Hail

The mathematical motivation underlying HAIL is illustrated by a transformation of the Bottom Set that more clearly exposes the component tasks of deductive and abductive reasoning. The Refined Bottom Set defined earlier certainly reveals the abductive and deductive components, but because it deliberately excludes vacuous literals, the Refined Bottom Set is not equivalent to the Bottom Set. For semantical purposes, it is desirable, at least initially, to obtain a logically equivalent reformulation of the Bottom Set that reveals this same separation into abductive and deductive components. It turns out that this can be achieved with relative ease by replacing the *Standardise* procedure by a simpler procedure, called *Normalise* in Definition 37 below, that simply leaves out the process of incapitation. Note that if e is a negative clause, then ϵ is read as the empty-clause.

Definition 37 (Normalise). Given a Horn theory B and a Horn clause e, let σ be a Skolemising substitution for both B and e, then the *Normalisation of B and e*, denoted $Normalise(B,e)$, is the pair $\langle \mathcal{B}, \epsilon \rangle$ defined as follows:

$$Normalise(B,e) = \langle \mathcal{B}, \epsilon \rangle \text{ where } \mathcal{B} = B \cup e_\sigma^- \text{ and } \epsilon = e_\sigma^+$$ ♦

Proposition 5.1.1 then provides a logically equivalent reformulation of the Bottom Set that applies in the case B and e are Horn. This reformulation shows that the computation of negative literals ($\neg\delta$) and positive literals (α) of the Bottom Set can be viewed as distinct tasks of deduction and abduction from the normalisation of B and e.

Proposition 5.1.1. *Let $\mathcal{B}$ and ϵ be the result of normalising a Horn theory B and a Horn clause e where $B \not\models e$. Let $\alpha, \delta \in \mathcal{GA}$ denote ground atoms. As usual, let the operators $\bigwedge$ and $\bigvee$ denote respectively the conjunction and disjunction of a set of formulae (sets of atoms in this case). Then*

$$Bot(B,e) \equiv \bigwedge\{\delta \in \mathcal{GA} \mid \mathcal{B} \models \delta\} \rightarrow \bigvee\{\alpha \in \mathcal{GA} \mid \mathcal{B} \wedge \alpha \models \epsilon\}$$

Proof. By Definition 16, $Bot(B,e) = \{L \mid B \wedge \overline{e} \models \neg L\}$. Partitioning into positive literals α and negative literals $\neg\delta$, this set can be rewritten as the union of the two sets: (i) $\{\alpha \mid B \wedge \overline{e} \models \neg\alpha\}$ and (ii) $\{\neg\delta \mid B \wedge \overline{e} \models \delta\}$. The proof is then by cases, according to whether e is a definite clause or a negative clause.

Case 1: Let e be the definite clause $e = \{E_0, \neg E_1, \ldots, \neg E_n\}$. Then set (i) is equal to $\{\alpha \mid B \wedge \neg E_0\sigma \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \neg\alpha\}$, which can be written $\{\alpha \mid \mathcal{B} \wedge \neg\epsilon \models \neg\alpha\}$, and is equal to $\{\alpha \mid \mathcal{B} \wedge \alpha \models \epsilon\}$. Set (ii) is equal to $\{\neg\delta \mid B \wedge \neg E_0\sigma \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \delta\}$, which can be written as $\{\neg\delta \mid \mathcal{B} \wedge \neg\epsilon \models \delta\}$, and this is now shown to be equal to $\{\neg\delta \mid \mathcal{B} \models \delta\}$ using the following argument. If δ is any atom such that $\mathcal{B} \models \delta$ then $\mathcal{B} \wedge \neg\epsilon \models \delta$ by monotonicity. Therefore $\{\neg\delta \mid \mathcal{B} \wedge \neg\epsilon \models \delta\} \supseteq \{\neg\delta \mid \mathcal{B} \models \delta\}$. If $\mathcal{B} \wedge \neg\epsilon \models \delta$ then $\mathcal{B} \wedge \neg\epsilon \wedge \neg\delta \models \bot$. Now, by the completeness of Hyper-resolution [CL73], there is a Hyper-resolution refutation from the clauses of $\mathcal{B} \cup \{\neg\epsilon\} \cup \{\neg\delta\}$, in which the electrons are $E_1\sigma, \ldots, E_n\sigma$ and any facts in $\mathcal{B}$. And since the nuclei $\neg\epsilon$ and $\neg\delta$ are negative unit clauses, they can be used only once (if at all) to derive the empty-clause in the very last step of the refutation. But suppose $\neg\epsilon$ is used, then $\neg\delta$ cannot be used, and so there is a Hyper-resolution refutation from the clauses of $\mathcal{B} \cup \{\neg\epsilon\}$, which means that $\mathcal{B} \wedge \neg\epsilon \models \bot$ by the soundness of Hyper-resolution, and so $\mathcal{B} \models \epsilon$. But this is equivalent to $B \models e$, which is a contradiction. Therefore $\neg\epsilon$ is not used, and so there is a Hyper-resolution refutation from the clauses of $\mathcal{B} \cup \{\neg\delta\}$, which means that $\mathcal{B} \models \delta$ by the soundness of Hyper-resolution. Therefore $\{\neg\delta \mid \mathcal{B} \wedge \neg\epsilon \models \delta\} \subseteq \{\neg\delta \mid \mathcal{B} \models \delta\}$. Hence $\{\neg\delta \mid \mathcal{B} \wedge \neg\epsilon \models \delta\} = \{\neg\delta \mid \mathcal{B} \models \delta\}$.

Case 2: Let e be the negative clause $e = \{\neg E_1, \ldots, \neg E_n\}$. Then set (i) is equal to $\{\alpha \mid B \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \neg\alpha\}$, which is equal to $\{\alpha \mid B \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \wedge \alpha \models \bot\}$ and can be written $\{\alpha \mid \mathcal{B} \wedge \alpha \models \epsilon\}$ as $\epsilon = \square$ whenever e is negative. Set (ii) is equal to $\{\neg\delta \mid B \wedge E_1\sigma \wedge \ldots \wedge E_n\sigma \models \delta\}$, which can be written $\{\neg\delta \mid \mathcal{B} \models \delta\}$.

In both cases $Bot(B,e) = \{L \mid B \wedge \overline{e} \models \neg L\} = \{\neg\delta \mid B \wedge \overline{e} \models \delta\} \cup \{\alpha \mid B \wedge \overline{e} \models \neg\alpha\} = \{\neg\delta \mid \mathcal{B} \models \delta\} \cup \{\alpha \mid \mathcal{B} \wedge \alpha \models \epsilon\}$. Since the clause $Bot(B,e)$ represents the disjunction of its literals, it is therefore logically equivalent to the formula $Bot(B,e) \equiv \bigwedge\{\delta \in \mathcal{GA} \mid \mathcal{B} \models \delta\} \rightarrow \bigvee\{\alpha \in \mathcal{GA} \mid \mathcal{B} \wedge \alpha \models \epsilon\}$. ∎

Proposition 5.1.1 shows that the atoms $\delta \in Bot^{-}(B,e)$ are those ground atoms that may be *deduced* from the normalised background $\mathcal{B}$, and that the atoms $\alpha \in Bot^{+}(B,e)$ are those ground atoms that may be *abduced* from $\mathcal{B}$ given as goal the normalised example ϵ. This has two important implications. First, the incompleteness of Progol5 identified in Section 3.3.3 can be avoided by replacing the STARTSET routine with an abductive procedure for deriving single atom hypotheses α. Second, the semantics of Bottom Generalisation can be extended, and the Progol5 proof procedure can be further generalised, by exploiting abductive hypotheses with multiple atoms, as shown in the next two subsections.

5.1.1 Kernel Generalisation and Kernel Set Subsumption

Motivated by a clear analogy with ALP, the equivalent formulation of the Bottom Set obtained in Proposition 5.1.1 is now generalised by replacing the single atoms α by conjunctions of atoms Δ. As formalised in 38 below, the resulting formula will be called the *Kernel* of B and e.

Definition 38 (Kernel). Let $\mathcal{B}$ and ϵ be the result of normalising a Horn theory B and a Horn clause e such that $B \not\models e$. Then the *Kernel* of B and e, written $\mathcal{K}er(B,e)$, is the formula defined as follows:

$$\mathcal{K}er(B,e) = \bigwedge\{\delta \in \mathcal{GA} \mid \mathcal{B} \models \delta\} \rightarrow \bigvee\{\Delta \subseteq \mathcal{GA} \mid \mathcal{B} \wedge \Delta \models \epsilon\}$$

♦

As formalised in Definition 39 and Proposition 5.1.2 below, any formula that logically entails the Kernel is said to be *derivable* by Kernel Generalisation, and all such formulae entail e relative to B. NOTE: from this point on it is assumed that all theories denoted by the symbol H are free of Skolem constants.

Definition 39 (Kernel Generalisation). Let B be a Horn theory and e be a Horn clause such that $B \not\models e$. Then a Horn theory H is said to be *derivable by Kernel Generalisation* from B and e iff

$$H \models \mathcal{K}er(B,e)$$

♦

Proposition 5.1.2 (Soundness of Kernel Generalisation). *Let H and B be Horn theories and let e be a Horn clause such that $B \not\models e$. Then*

$$\textit{if } H \models \mathcal{K}er(B,e) \textit{ then } B \wedge H \models e$$

Proof. Assume $H \models \mathcal{K}er(B,e)$. For convenience, first let P and S abbreviate the following formulae: let $P = \bigwedge\{\delta \in \mathcal{GA} \mid \mathcal{B} \models \delta\}$ be the conjunction of all ground atoms entailed by $\mathcal{B}$, and let $S = \bigvee\{\Delta \in \mathcal{AGA} \mid \mathcal{B} \wedge \Delta \models \epsilon\}$ be the disjunction of the conjunctions of ground atoms that together with $\mathcal{B}$ entail ϵ. Then observe that (i) $\mathcal{B} \models P$ as each conjunct δ of P is individually entailed by $\mathcal{B}$, and (ii) $\mathcal{B} \wedge S \models \epsilon$ as together with $\mathcal{B}$, each individual conjunct Δ of S entails ϵ, and (iii) $H \models P \rightarrow S$ by Definition 38 and assumption above. Let $\mathcal{M}$ be any model of $\mathcal{B} \wedge H$. Then because $\mathcal{M}$ is a model of $\mathcal{B}$ it is also a model of P, using (i). And because $H \models \mathcal{K}er(B,e)$ and $\mathcal{M}$ is a model of H and P, it is also a model of S, using (iii). And because $\mathcal{M}$ is a model of $\mathcal{B}$ and S, it is also a model of ϵ, using (ii). Therefore $\mathcal{B} \wedge H \models \epsilon$, which is equivalent to $B \wedge H \models e$. ∎

For computational purposes it is convenient to introduce a refinement of the Kernel, called a *Kernel Set*. Informally, a Kernel Set $\mathcal{K}$ is a Horn clause representation of some part of a Kernel $\mathcal{K}er(B,e)$. As shown in Definition 40 below, the head atoms α of $\mathcal{K}$ constitute one conjunction Δ from the consequent of $Ker(B,e)$, and the body atoms δ_i^j of $\mathcal{K}$ represent a selection of δ's from the antecedent of $\mathcal{K}er(B,e)$. And, as shown in Definition 41 and Proposition 5.1.3 below, a sound inductive inference procedure called *Kernel Set Subsumption* is obtained by searching for *theories* H that clausally subsume a Kernel Set of B and E.

Definition 40 (Kernel Set). Let $\mathcal{B}$ and ϵ be the result of normalising a Horn theory B and a Horn clause e. Then a Horn theory $\mathcal{K}$ is said to be *a Kernel Set* of B and e iff

$$\mathcal{K} = \left\{ \begin{array}{l} \alpha_1 \,\text{:-}\, \delta_1^1, \ldots\ldots\ldots\ldots, \delta_1^{m(1)} \\ \qquad\qquad \vdots \\ \alpha_i \,\text{:-}\, \delta_i^1, \ldots, \delta_i^j, \ldots, \delta_i^{m(i)} \\ \qquad\qquad \vdots \\ \alpha_n \,\text{:-}\, \delta_n^1, \ldots\ldots\ldots\ldots, \delta_n^{m(n)} \end{array} \right\}$$

where $m(i) \geq 0$ denotes the number of body atoms in the i^{th} clause, and $\alpha_i \in \mathcal{GA}$ denotes the head atom of the i^{th} clause, and $\delta_i^j \in \mathcal{GA}$ denotes the j^{th} body atom of the i^{th} clause, and $\mathcal{B} \cup \{\alpha_1, \ldots, \alpha_n\} \models \epsilon$ and $\mathcal{B} \models \delta_i^j$ for all i, j such that $1 \leq i \leq n$ and $0 \leq j \leq m(i)$.

Definition 41 (Kernel Set Subsumption). Let $\mathcal{K}$ be a Kernel Set of a Horn theory B and a Horn clause e such that $B \not\models e$. Then a Horn theory H is said to be *derivable by Kernel Set Subsumption* from B and e iff $H \sqsupseteq \mathcal{K}$ ♦

Proposition 5.1.3 (Soundness of Kernel Set Subsumption). *Let $\mathcal{K}$ be a Kernel Set of a Horn theory B and a Horn clause e such that $B \not\models e$. Then*

$$\textit{if } H \sqsupseteq \mathcal{K} \textit{ then } H \models \mathcal{K}er(B, e)$$

Proof. Assume $H \sqsupseteq \mathcal{K}$. For convenience, first let P, Q, R and S abbreviate the following formulae: let $P = \bigwedge\{\delta \in \mathcal{GA} \mid \mathcal{B} \models \delta\}$ be the conjunction of all ground atoms entailed by $\mathcal{B}$, let $Q = \bigwedge\{\delta_1^1, \ldots, \delta_i^j, \ldots, \delta_n^{m(n)}\}$ be the conjunction of all body atoms of $\mathcal{K}$, let $R = \bigwedge\{\alpha_1, \ldots, \alpha_n\}$ be the conjunction of all head atoms of $\mathcal{K}$, and let $S = \bigvee\{\Delta \in \mathcal{AGA} \mid \mathcal{B} \cup \Delta \models \epsilon\}$ be the disjunction of the conjunctions of ground atoms that together with $\mathcal{B}$ entail ϵ. Then observe that (i) $P \models Q$ as the conjuncts δ_i^j of Q are included among the conjuncts δ of P, and (ii) $R \models S$ as the conjunction R is one of the disjuncts Δ in S, and (iii) $\mathcal{K} \models Q \rightarrow R$, as any model of $\mathcal{K}$ that satisfies every body atom, must also satisfy every head atom, and (iv) $H \models \mathcal{K}$ by definition of θ-subsumption and the assumption above. Now assume $\mathcal{M}$ is a model of H, and let $\mathcal{M}$ be any model of P. Then $\mathcal{M}$ is also a model of Q, using (i). And because $\mathcal{M}$ is a model of H, it is also a model of $\mathcal{K}$, using (iv). And because $\mathcal{M}$ is a model of $\mathcal{K}$ and Q, it is also a model of R, using (iii). And because $\mathcal{M}$ is a model of R, it is also a model of S, using (ii). Therefore $H \models P \rightarrow S$, or equivalently $H \models \mathcal{K}er(B, e)$ by Definition 38. ▮

Propositions 5.1.2 and 5.1.3 above, show that Kernel Set Subsumption is a sound method of inductive generalisation. Proposition 5.1.4 below, shows that Kernel Set Subsumption is a strict extension of Bottom Generalisation.

Proposition 5.1.4 (Kernel Set Subsumption extends Bottom Generalisation). *Let B be a Horn theory B and e a Horn clause such that $B \not\models e$. Then the set of hypotheses KSS derivable by Kernel Set Subsumption from B and e, strictly includes the set of hypotheses BG derivable by Bottom Generalisation from B and e.*

Proof. First show that $KSS \supseteq BG$. If the Horn clause h is derivable from B and e by Bottom Generalisation, then $h \succcurlyeq Bot(B, e)$ by Definition 18, and therefore $h\sigma \subseteq Bot(B, e)$ for some substitution σ. And by Proposition 5.1.1 $h\sigma = \alpha \,\text{:-}\, \delta_1, \ldots, \delta_n$ where $\mathcal{B} \wedge \alpha \models \epsilon$ and $\mathcal{B} \models \delta_j$ for all $0 \leq j \leq n$. Therefore the Horn theory $H = \{h\}$ is derivable by Kernel Set Subsumption, using the Kernel Set $\mathcal{K} = \{\alpha \mathrel{:-} \delta_1, \ldots, \delta_n\}$. Thus $KSS \supseteq BG$.

Now show $KSS \neq BG$. Let $p/0$ and $q/1$ be predicates and a and b be constants, and define:

$$B = \{p \mathrel{:-} q(a), q(b)\} \qquad e = p \qquad h = q(X)$$

Then the hypothesis $h = q(X)$ is *not* derivable by Bottom Generalisation, as it does not θ-subsume the Bottom Set $Bot(B, e) = \{p\}$. But the hypothesis $h = q(X)$ *is* derivable by Kernel Set Subsumption as it does clausally subsume the Kernel Set $\mathcal{K} = \{q(a), q(b)\}$. Thus $KSS \neq BG$, and hence $KSS \supset BG$. ▮

Note that a refined inference method is obtained from *Kernel Set Subsumption* by replacing the *Normalise* transformation by the *Standardise* transformation in Definition 40 – giving the notion of a *Refined Kernel Set* – and by replacing H by $H_{\vec{\mathrm{ff}}}$ prior to testing for subsumption in Definition 41 – giving the notion of *Refined Kernel Set Subsumption.* This refined approach has the advantage of computing fewer vacuous literals.

This section now concludes with the observation that Kernel Set Subsumption is related to an extension of Plotkin's c-derivation, which is formalised in Definition 42 below.

Definition 42 (K-derivation). Define a *K-derivation* of a clause e from a clausal theory B *with respect to* a clausal theory H, as a resolution derivation of e from $B \cup H$, in which any clause in H is used at most once. A K-derivation of the empty-clause is called a *K-refutation.* ♦

Thus, a K-derivation[1] is like a c-derivation, except that now there is a *set* of clauses, each of which may be used at most once. The c-derivation is therefore a special case of the K-derivation in which this set is a singleton. With this new definition in mind, it remains to show that a theory $\mathcal{K}$ is a Kernel Set of B and e only if there exists a K-refutation from $B \cup \overline{e}$ with respect to $\mathcal{K}$.

5.2 A Generic Proof Procedure for Hail

This section introduces a generic proof procedure for Kernel Set Subsumption that integrates abductive, deductive and inductive reasoning within a general framework. The purpose of this section is to show how multiple-atom abductive hypotheses can be used productively within a cycle of learning that generalises Progol5.

The intuition underlying the HAIL proof procedure is illustrated in Figures 5.1, which provides a conceptual view of the HAIL learning cycle. In common with Progol5, this cycle consists of an outer Cover-Set loop (Steps 1 and 5) and three sub-routines that constitute the abductive (Step 2), deductive (Step 3) and inductive (Step 4) phases, respectively. Given as input finite Horn theories B, E^+ and E^-, and a set of mode-declarations M, the objective

[1] Note that an equivalent notion, called Input-Quota or IQ-Resolution, can be defined as follows: Given a set of clauses T and a partial function $\mathcal{Q}$ from the clauses in T to the natural numbers $\mathcal{N}$, then an IQ-derivation of a clause C from $\langle T, \mathcal{Q} \rangle$ is a resolution derivation of C from T, in which each clause c is used as an input clause at most $\mathcal{Q}(c)$ times, whenever $\mathcal{Q}(c)$ is defined, and any number of times $\mathcal{Q}(c)$ is not defined. Informally, the function $\mathcal{Q}$ represents a 'quota' for the clauses in T. Note that there is a straightforward correspondence between IQ-derivations and K-derivations: namely an IQ-derivation in which clause c has a quota of n, is simulated by a K-derivation with respect to n variants of c, and vice-versa.

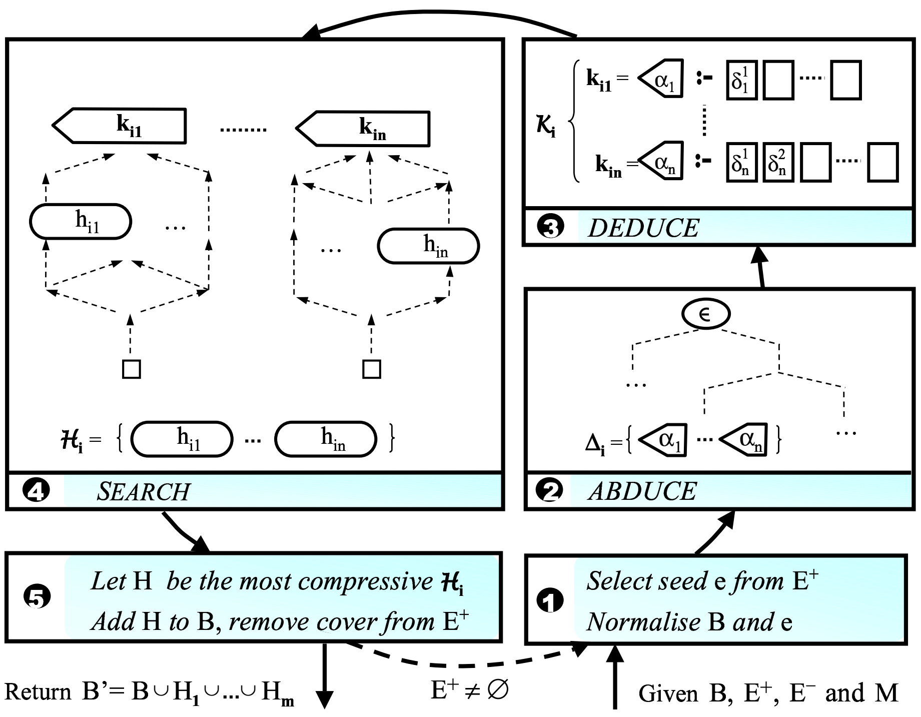

Figure 5.1: A Conceptual View of the HAIL Learning Cycle

```
Begin HAIL
    Input                given $B$, $E^+$, $E^-$, $M$
                           assert $Consistent(B \cup E^+ \cup E^-)$
                           let $E^+ = E^+ - Cover(B, E^+)$
    Cover-Set-Loop         while $E^+ \neq \emptyset$
    select seed                select seed example $e \in E^+$
    standardise                let $\langle \mathcal{B}, \epsilon \rangle = Standardise(B, e)$
    Abduction                  let $\mathcal{A} = ABDUCE(\mathcal{B}, \epsilon, M^+)$
    Deduction                  for each abduced hypothesis $\Delta_i \in \mathcal{A}$
                                   for each abduced atom $\alpha_j \in \Delta_i$
                                       let $k_{ij} = DEDUCE(\mathcal{B}, \alpha_j, M^-)$
                                   let $\mathcal{K}_i = \bigcup_j \{k_{ij}\}$
    Induction                      let $\mathcal{H}_i = SEARCH(\mathcal{K}_i, B, E^+, E^-, M)$
    best hypothesis            let $H = \mathcal{H}_i$ with greatest $Compression$
    assert hypothesis          let $B = B \cup H$
    remove cover               let $E^+ = E^+ - \{e \in E^+ \mid B \models e\}$
    Output               return $B$
End HAIL
```

Figure 5.2: An Overview of the HAIL Learning Cycle

of HAIL is to return an augmented background knowledge $B' = B \cup H_1 \cup \ldots \cup H_n$ which entails E^+, is consistent with E^-, and where each theory H is maximally compressive and compatible with M. It is initially assumed that E^+ is non-empty and contains no clause that is entailed by B. On every iteration of the cycle at least one clause is removed from E^+ and at least one clause is added to B.

The CoverSet loop begins (Step 1) by selecting from E^+ a seed example e, which is standardised with B, giving theory $\mathcal{B}$ and atom ϵ. An abductive procedure is then used (Step 2) to find explanations $\Delta_i = \{\alpha_1, \ldots, \alpha_n\}$ of goal ϵ from theory $\mathcal{B}$. By definition, each explanation is a set of implicitly conjoined ground atoms such that $\mathcal{B} \wedge \Delta_i \models \epsilon$. Any abductive procedure can be used, but for the purposes of illustration Figure 5.1 depicts a tree-like computation representing the ASLD procedure of Kakas and Mancarella [KM90]. Abduced atoms α_j are shown as tapered squares, the goal ϵ is shown as an oval, and the theory $\mathcal{B}$ is implicit.

Every n-atom hypothesis $\Delta_i = \{\alpha_1, \ldots, \alpha_n\}$ abduced in Step 2 is used in Step 3 to form an n-clause Kernel Set $\mathcal{K}_i = \{k_{i1}, \ldots, k_{in}\}$, with each atom α_j becoming the head of exactly one clause k_{ij}. To every head atom α_j is adjoined a set of body atoms δ_i^j, shown as squares in Figure 5.1, each of which is determined by a deductive procedure that computes ground atomic consequences of $\mathcal{B}$. The resulting Kernel Set $\mathcal{K}_i$ is then generalised (Step 4) by constructing a Horn theory $\mathcal{H}_i$ that includes at least one clause h_{ij} from the θ-subsumption lattice of each Kernel clause k_{ij}. Figure 5.1 shows the clauses h_{i1} and h_{in} (rounded rectangles) selected from the θ-subsumption lattices (dotted arrows) of the Kernel clauses k_{i1} and k_{in} (tapered rectangles). In general, the same clause may be selected from several lattices, as in the example used in Proposition 5.1.4.

The hypotheses constructed by HAIL should be compatible with the given language bias, and they should be maximally compressive in the sense of covering the greatest number of remaining positive examples while containing the fewest number of literals. Therefore, the abductive and search procedures are required return hypotheses that are minimal in the sense that no subset is also a hypothesis, and, in practice, all three procedures will make use of the mode-declarations M. In this way, the most compressive hypothesis $\mathcal{H}_i$ is determined for each Kernel Set $\mathcal{K}_i$ resulting from some explanation Δ_i. In step 5, the most compressive such hypothesis, H, is then asserted into B, and any covered examples are removed from E^+. The cycle is repeated until E^+ is empty, whereupon the augmented background B' is returned.

A more formal overview of the HAIL learning cycle is shown in Figure 5.2, which represents the intuition presented above in terms of the generic procedures *ABDUCE*, *DEDUCE* and *SEARCH*. aborts if the inputs are inconsistent, as in this case no solution can exist;

Given as input B, E^+, E^- and M, HAIL begins by verifying that the inputs are consistent – as otherwise no solution exists; and then it removes from E^+ any examples already covered by B – as these require no hypothesis. The first seed example is then selected and standardised, giving $\mathcal{B}$ and ϵ. From the theory $\mathcal{B}$ and the goal ϵ, *ABDUCE* computes a set $\mathcal{A} = \{\Delta_1, \ldots, \Delta_p\}$ of explanations, each of which is an implicitly conjoined set $\Delta_i = \{\alpha_1, \ldots, \alpha_n\}$ of ground atoms compatible with the head-declarations M^+ in M, and is such that $B \wedge \Delta_i \models e$.

In the outer for-loop, each explanation $\Delta_i \in \mathcal{A}$ is processed in turn. In the inner for-loop, each atom $\alpha_j \in \Delta_i$ becomes the head of a clause k_{ij} to which *DEDUCE* adjoins a set body atoms, each of which is a ground atomic consequence of $\mathcal{B}$ compatible with the

```
Begin Algorithm 5.3.1 (HAIL).
  Input               given $B$, $E^+$, $E^-$, $M$
  check consist.        assert $Consistent(B \cup E^+ \cup E^-)$
  remove cover          let $E^+ = E^+ - Cover(B, E^+)$
  Cover-Set-Loop        while $E^+ \neq \emptyset$
  select seed               choose seed example $e \in E^+$
  standardise               let $\langle \mathcal{B}, \epsilon \rangle = Standardise(B, e)$
  Abduction                 let $\mathcal{A} = A\text{SLD}(T\text{-FORM}(\mathcal{B}, \epsilon, E^-, M^+))$
  Deduction                 for each abduced hypothesis $\Delta_i \in \mathcal{A}$
                                for each abduced atom $\alpha_j \in \Delta_i$
                                    for some head-dec $d \in M^+$ such that $schema(d) \succcurlyeq \alpha_j$
                                        let $k_{ij} = B\text{OTTOM}S\text{ET}(\mathcal{B}, \alpha_j, d, M^-)$
                                let $\mathcal{K}_i = \bigcup_j \{k_{ij}\}$
  Induction                     let $H_i = M\text{-SEARCH}(\mathcal{K}_i, B, E^+, E^-, M)$
  assert best hyp.          add to $B$ the $H_i$ with greatest $Compression$
  remove cover              let $E^+ = E^+ - Cover(B, E^+)$
  Output              return $B$
End Algorithm (HAIL)
```

body-declarations M^- in M. The Kernel Set $\mathcal{K}_i$ formed of the union of the clauses k_{ij} is then generalised by SEARCH, which determines the most compressive theory $\mathcal{H}_i$ that clausally subsumes $\mathcal{K}_i$ and is compatible with M. The most compressive theory obtained in this way is then added to B, and any newly covered examples are removed from E^+.

5.3 A Concrete Proof Procedure for Hail

This section introduces a concrete proof procedure for refined Kernel Set Subsumption that is based on the Bottom-based Cover-Set methodology described in Algorithm 3.2.18, but is able to find hypotheses that lie outside the semantics of straightforward Bottom Generalisation. The procedure refines that described above by providing concrete algorithms for the abductive, deductive and inductive phases.

Full ASLD is used in the abductive phase, but in order to guarantee the computed atoms are compatible with the mode-declarations, the T-FORM algorithm pre-processes the inputs to the ASLD procedure. The standard Progol5 BOTTOMSET routine is used unaltered in the deductive phase. The M-SEARCH routine is used to generalise the computed Kernel Sets. It remains only to describe the novel components T-FORM and M-SEARCH.

5.3.1 T-FORM

One practical problem associated with the integration of ALP and ILP technologies arises from the diversity of mechanisms used for representation the of language bias. Whereas ASLD employs abducible predicates, Progol5 utilises mode-decs. Of these two approaches, mode-decs are strictly more expressive in the sense that they not only determine which predicates may be appear in the heads and bodies of hypothesised clauses, but also which term-structures may appear as the arguments to those predicates.

```
Begin Algorithm 5.3.2 (T-FORM).
  Input            given $\mathcal{B}$, $\epsilon$, $E^-$, $M^+$
                     let $T = \mathcal{B}$, $g = \epsilon$, $IC = \emptyset$ and $A = \emptyset$
  Auxiliary          for each predicate $p/n$ such that $p = pred(m)$ for some $m \in M^+$
  predicates             let $p^\delta/n$ and $p^\tau/n$ be fresh predicate symbols
                         if $p$ is defined in $T$ then
                             let $A = A \cup \{p^\delta\}$
  bridging clause            let $T = T \cup \{p(X_1,\ldots,X_n) \text{:-} p^\delta(X_1,\ldots,X_n)\}$
                             let $IC = IC \cup \{ \text{:-} p^\delta(X_1,\ldots,X_n), \backsim p^\tau(X_1,\ldots,X_n)\}$
                         else
                             let $A = A \cup \{p\}$
                             let $IC = IC \cup \{ \text{:-} p(X_1,\ldots,X_n), \backsim p^\tau(X_1,\ldots,X_n)\}$
  Type               for each head-dec $m \in M^+$
  constraints            let $T = T \cup \{schema^\tau(m) \text{:-} type(m)\}$
  Domain             for each denial $\text{:-} a_1,\ldots,a_m \in E^-$
  constraints            if $pred(a_j) \notin A$ for all $1 \le j \le m$ then
                             let $IC = IC \cup \{ \text{:-} a_1,\ldots,a_m, p(X_1,\ldots,X_n) \mid p/n \in A\}$
                         else
                             let $IC = IC \cup \{ \text{:-} a_1,\ldots,a_m\}$
  Output           return $\langle T, g, IC, A\rangle$
End Algorithm (T-FORM)
```

If abduced atoms are to participate in the heads in the heads of induced clauses, then the abductive procedure must be constrained so as to return only atoms that conform to the inductive language bias. More concretely, the ASLD procedure must be adapted to return only those atoms that are compatible with a given set of head-decs. The T-FORM routine shown in Algorithm 5.3.2 ensures precisely this, by encoding the inductive language bias as abductive integrity constraints and additional program rules containing new predicates.

T-FORM combines the encoding of language bias together with the pre-processing necessary to satisfy the restrictions imposed by ASLD; namely that all abducible predicates be undefined in T and all integrity constraints mention one abducible. In addition T-FORM encodes negative examples as integrity constraints (since all hypotheses must be consistent with the negative examples). T-FORM therefore maps an inductive context into an abductive context for the purpose of abducing head atoms to be used in the construction of Kernel Sets.

The inputs to the T-FORM algorithm are determined by the inductive problem and consist of the standardised background knowledge $\mathcal{B}$ and seed example ϵ, negative examples and domain integrity constraints E^-, and head-decs M^+. The output is an abductive context $\langle T, g, IC, A\rangle$ satisfying the restrictions of the ASLD procedure, and in which are encoded the necessary language and domain constraints to ensure that all abduced hypotheses are compatible with M^+ and consistent with E^-.

The T-FORM algorithm begins by initialising $T = \mathcal{B}$, $g = \epsilon$ and $IC = \emptyset$. Since ASLD is being used to abduce those atoms that will participate in the *heads* of induced clauses, each head-dec d is examined in turn to determine which predicates p should be considered abducible. But because ASLD requires that no abducible predicate be defined in the theory, auxiliary predicates p^δ are instead added to $\mathcal{A}$, and bridging clauses $p(X_1,\ldots,X_n) \text{:-} p^\delta(X_1,\ldots,X_n)$ are added to T. Conversely, in each hypotheses Δ formed by ASLD, every auxiliary predicate p^δ is replaced by the corresponding base predicate p.

To facilitate the encoding of the inductive language constraints on the arguments of p, another auxiliary predicate p^τ is introduced. Intuitively, the call $p^\tau(t_1, \ldots, t_n)$ succeeds iff the atom $p(t_1, \ldots, t_n)$ conforms to the required language bias. If it does not, and an attempt is made to abduce $p^\delta(t_1, \ldots, t_n)$, the constraint $:\!\!-\, p^\delta(X_1, \ldots, X_n), \backsim p^\tau(X_1, \ldots, X_n)$ will be violated and the current hypothesis will be aborted. The language bias is extracted from the mode-dec using the *schema* and *type* functions described in definition 8 and added to T.

General domain integrity constraints and negative examples can be incorporated into the abductive context almost directly. But because ASLD requires that all integrity constraints mention at least one abducible predicate, a simple transformation must be applied. To force ASLD to check each domain constraint whenever *any* atom is abduced, several integrity constraints are formed by appending to the original constraints the header of each abducible predicate in turn. Every abducible is now guaranteed to resolve with exactly one of these, leaving a resolvent to be checked that is identical to the original constraint. This process is correct providing that the original constraint is not initially violated, and this will be guaranteed by the cover-set-loop.

Example 5.3.3. The intuition that underlies the T-FORM algorithm is illustrated by the contexts shown in Figures 3.1 and 4.1. Clearly T and g correspond to the background knowledge B and the seed example e. The integrity constraints $ic1$ and $ic2$ can now intuitively be seen to encode the negative example and head-dec.

First consider the head-dec $\mathsf{modeh}[fries(+bistro)]$ which states that in the MSH all arguments of the predicate $fries/1$ must be of type *bistro*. This is encoded in the integrity constraint $:\!\!-\, fries(V), \backsim bistro(V)$ which states that it must not be the case that any argument to $fries$ fails to be a known *bistro*.

Now consider the negative example $:\!\!-\, meal(theRitz)$ which states that it must not be the case that a meal was had in *theRitz*. It is appropriate to treat this restriction as an integrity constraint, but because *meal* is not abducible the additional atom $fries(W)$, corresponding to the only abducible predicate, must be added.

Formally, the application of the T-FORM algorithm to the inductive context of figure 3.1 would result in the clauses $fries(X) :\!\!-\, fries^\delta(X)$ and $fries^\tau(X) :\!\!-\, bistro(X)$ being added to T, and the integrity constraints $:\!\!-\, fries^\delta(X), \backsim fries^\tau(X)$ and $:\!\!-\, meal(theRitz), fries^\delta(X)$ being added to IC. The hypothesis constructed by ASLD before removal of auxiliary predicates is $fries^\delta(mcDonalds)$.

5.3.2 M-SEARCH

M-SEARCH takes as input a finite list of clauses $\mathcal{K}$ (Kernel Set), and a Horn mode inductive context consisting of the Horn clause theories B (background knowledge), E^+ (positive examples), and E^- (negative examples), and a set of mode-decs M (language bias). The task of M-SEARCH is to compute the most compressive hypothesis H such that every clause k in $\mathcal{K}$ is subsumed by at least one clause in H. And this it achieves by selecting one[2] representative from each θ-subsumption sub-lattice $[\square, k]$.

[2]A less naive approach is considered in Subsection 5.3.4.i, but for the purposes of exposition is appropriate to consider a more simplistic approach first.

Begin Algorithm 5.3.4 (M-SEARCH).

```
Input            given $\mathcal{K}$, $B$, $E^+$, $E^-$ and $M$
Base               if $\mathcal{K} = [\,]$ then
case                   $\langle bestH, bestS\rangle = \langle \blacksquare, Compression(B, \blacksquare, E^+, E^-)\rangle$ [a]
General            else
case                   let $\langle bestH, bestS\rangle = \langle \blacksquare, -\infty\rangle$
                       let $hs = [head(\mathcal{K})]$ [b]
                       let $visited = \emptyset$
                       while $hs \neq [\,]$
                           let $\langle newH, newS\rangle = \langle \blacksquare, -\infty\rangle$
                           let $h = head(hs)$
                           if $h \notin visited$ then
                               if $h \notin \mathcal{H}_M$ then [c]
open sub-tree                      let $hs = hs{+}{+}\gamma(h)$ [d]
                               else
recursion step                     let $\langle newH, newS\rangle = M\text{-SEARCH}(tail(\mathcal{K}), B \cup \{h\}, E^+, E^-, M)$
                                   if $newS > -\infty$ then
open sub-tree                          let $hs = hs{+}{+}\gamma(h)$
                                       if $newS - |h| > bestS$ then
                                           let $\langle bestH, bestS\rangle = \langle newH \cup \{h\}, newS - |h|\rangle$
                               let $visited = visited \cup \{h\}$
                           let $hs = tail(hs)$
Output           return $\langle bestH, bestS\rangle$
```

End Algorithm (M-SEARCH)

[a] n.b. Compression is definied in Definition 9.
[b] n.b. For an alternative see Subsection 5.3.4.
[c] n.b. $\mathcal{H}_M$ is the hypothesis space determined by M.
[d] n.b. $\gamma(h)$ are immediate generalisations of h under $\succeq$.

Note that it is not viable to generalise each MSH individually because the presence of the others is needed in order to cover the seed example and to check consistency. Note also that when searching for the most compressive hypothesis, the inter-dependency between the sub-lattices potentially results in a large computational overhead that in practice must be ameliorated with suitable control strategies.

Algorithm 5.3.4 performs a naive specific-to-general search through the sub-lattices determined by the Kernel Set. Each hypothesis considered by the algorithm, is obtained picking one representative from each lattice, and assessing the compressivity of that hypothesis, with respect to the examples and background knowledge. Computational effort is conserved by pruning inconsistent hypotheses from the search space, and by ignoring hypotheses which do not conform to the given language bias.

Given a non-empty list $\mathcal{K}$ of clauses (constituting a Kernel Set), Algorithm 5.3.4 picks the first clause in the list and begins a breadth-first specific-to-general search of the θ-subsumption sub-lattice bounded by the chosen clause and the empty-clause. During the search a list *hs* is maintained of the clauses still to be investigated, and a list *visited* is maintained of the clauses that have already been investigated. Initially *hs* is set to the chosen kernel clause, and *visited* is empty.

The best hypothesis found so far is stored in a variable $bestH$, and its corresponding score is stored in the variable $bestS$. The variable $bestH$ contains a clausal theory and is initially empty. The variable $bestS$ contains a score and is initially set to $-\infty$, which is the lowest possible score. On each iteration a clause h is chosen from the head of hs and investigated. If h has already been visited then it is ignored. If h is not compatible with the hypothesis space $\mathcal{H}_M$ determined by the mode-decs M, then it is ignored, but only after its descendents $\gamma(h)$ are added to hs.

If the current clause h is being visited for the first time and it lies within the given language bias, then it is temporarily asserted into the background knowledge, and the optimal choices in the remaining sub-lattices are determined by recursing on the tail of $\mathcal{K}$. If the new score $newS$ returned by the recursive call is greater than $-\infty$ then a hypothesis $newH$ was constructed that is consistent with the background knowledge B (which may contain hypothesis clauses of any previous sub-lattices). Therefore, the best hypothesis which can be constructed using h is $newH \cup \{h\}$ with a corresponding compression of $newS - |h|$ (remembering to subtract the complexity of h).

If this newly determined score is the best so far, then the old pair $\langle bestH, bestS\rangle$ are overwritten with the new pair $\langle newH \cup \{h\}, newS - |h|\rangle$. The current clause h is marked as visited, and if a consistent hypothesis was constructed, then the immediate generalisations $\gamma(h)$ of h are added to hs. Once all generalisations have been exhausted, the algorithm returns the most compressive hypothesis that was constructed, together with the corresponding score.

In the base case one representative from each lattice has been added to B and the list $\mathcal{K}$ is now empty. All that remains is to return the compression score with respect to the positive and negative examples. The null-clause formally takes the place of the hypothesis argument, which as just stated, is already contained in the background knowledge.

5.3.3 Worked Examples

The HAIL proof procedure is now illustrated by means of two simple worked examples. The first example illustrates how HAIL is able to overcome the incompleteness due to the Progol5 STARTSET routine. The second example shows how HAIL is able to derive hypotheses outside the semantics of Bottom Generalisation.

Example 5.3.5 (Fast Food). This example shows how HAIL is able to overcome the incompleteness of Progol5 identified in Subsection 3.3.3. For convenience, the inductive context, which appeared previously in Figure 3.1, is repeated in Figure 5.3, and the corresponding HAIL learning cycle is illustrated in Figure 5.4.

Recall from Section 2.4 and Figure 2.1 that the background knowledge B describes a domain of three *bistro*s: md, bk and rz (abbreviating $mcDonalds$, $burgerKing$ and $theRitz$). To have a $meal$ in a $bistro$, it is sufficient to have $burger$ and $fries$; and a free $burger$ comes with every order of $fries$ at *bistro*s participating in a special $offer$. Observe that the positive examples E^+ state that a $meal$ has been eaten at both md and bk; and that the negative examples E^- state that a $meal$ has *not* been eaten at rz. The mode-declarations state that the atom $fries(X)$ may appear in the heads of hypothesised clauses; and the atom $offer(X)$ may appear in the bodies. Recall from Example 3.2.3 that hypothesis H is derivable by Bottom Generalisation using either $e = meal(md)$ or $e = meal(bk)$ as the seed example. Finally recall from Examples 3.3.6 and 3.3.8 that this hypothesis cannot be computed by Progol5.

As illustrated in Figure 5.4, HAIL solves Example 5.3.5 in the following way. In Step 1, the seed $e = meal(md)$ is selected and standardised, trivially giving $\mathcal{B} = B$ and $\epsilon = e$. In Step 2, given theory $\mathcal{B}$ and goal ϵ, ASLD abduces the hypothesis $\Delta = \{fries(md)\}$ containing the atom $\alpha = fries(md)$. In Step 3, this atom α becomes the head of a clause k, to which the body atom $offer(md)$ is added by BOTTOMSET. For efficiency BOTTOMSET replaces the constant md with the variable X, as required by the mode-declarations. In Step 4, the θ-subsumption lattice, bounded from above by the newly computed clause k, and from below by the empty clause $\Box$, is searched. The most compressive hypothesis is k itself – as all more general clauses are inconsistent with the negative example $\text{:-}\, meal(rx)$. In Step 5, the theory $H = \{fries(X) \text{:-}\, offer(X)\}$ is added to B, and because both positive examples are now covered, they are removed from E^+. The cycle terminates, returning the augmented background $B' = B \cup H$.

Example 5.3.6 (Academic). This example shows how HAIL is able to compute more than one clause in response to a single seed example, and to derive hypotheses outside the semantics of Bottom Generalisation. The inputs are shown in Figure 5.5 and the corresponding HAIL learning cycle is illustrated in Figure 5.6. This example concerns *students*, *lecturers* and *academics*.

The background knowledge B has been partitioned into domain knowledge and scenario knowledge. The former states that being *tired* and *poor* is enough to make anybody *sad*, and that *students* and *lecturers* are all *academics*. The latter states that there is one *student* called *oli* and two *lecturers* called *ale* and *kb*. The examples and mode-declarations are self-evident. It can be verified that the hypothesis H, which asserts that everybody is *tired* but that all *lecturers* are *poor*, is the shortest hypothesis that is compatible with the mode-declarations and explains the positive examples and is consistent with the negative examples. It can also be verified that H cannot be derived by Bottom Generalisation using either of the positive examples as a seed. This is because no literal with the predicates *tired* or *poor* is entailed by the complementary extension B_{e^*}, and so no such literal is contained in the $Bot(B, e)$, nor any clause derivable by Bottom Generalisation. In particular, observe, therefore, that Progol5 cannot compute H.

As illustrated in Figure 5.6, HAIL solves Example 5.3.6 in the following way. In Step 1, the seed $e = sad(ale)$ is selected and standardised, again giving $\mathcal{B} = B$ and $\epsilon = e$. In Step 2, ASLD abduces the hypothesis Δ containing the two atoms $\alpha_1 = tired(ale)$ and $\alpha_2 = poor(ale)$. In Step 3, α_1 and α_2 become the heads of two clauses k_1 and k_2, to which the body atoms $lecturer(ale)$ and $academic(ale)$ are added by BOTTOMSET. Note that for efficiency BOTTOMSET replaces the constant ale with the variable X, as required by the mode-declarations. Note also that, in general, different body atoms will be added to different clauses. Note finally that the two clauses k_1 and k_2 constitute a Kernel Set of B and e. In Step 4, one clause is chosen from each of the θ-subsumption lattices resulting from this Kernel Set. For ease of presentation the clauses in the θ-subsumption lattices have been written without brackets and only the first letter of each predicate symbol is shown. In Step 5, the most compressive hypothesis H consisting of the two clauses $tired(X)$ and $poor(X) \text{:-}\, lecturer(X)$ is added to B, and because both positive examples are now covered, they are removed from E^+. The cycle then terminates returning the augmented background $B' = B \cup H$.

Background Knowledge
$B = \left\{ \begin{array}{l} meal(X) \text{:-} fries(X),\ burger(X) \\ burger(Y) \text{:-} fries(Y),\ offer(Y) \end{array} \right\} \cup$ $\left\{ \begin{array}{l} bistro(md) \\ bistro(bk) \\ bistro(rz) \end{array} \right\} \cup \left\{ \begin{array}{l} offer(md) \\ offer(bk) \\ burger(rz) \end{array} \right\}$

Positive Examples	Negative Examples
$E^+ = \left\{ \begin{array}{l} meal(md) \\ meal(bk) \end{array} \right\}$	$E^- = \{\ \text{:-} meal(rz)\ \}$

Head-Declarations	Body-Declarations
$M^+ = \{\ \mathsf{modeh}[fries(+bistro)]\ \}$	$M^- = \{\ \mathsf{modeb}[offer(+bistro)]\ \}$

Hypothesis
$H = \{\ fries(Z) \text{:-} offer(Z)\ \}$

Figure 5.3: Fast Food Example

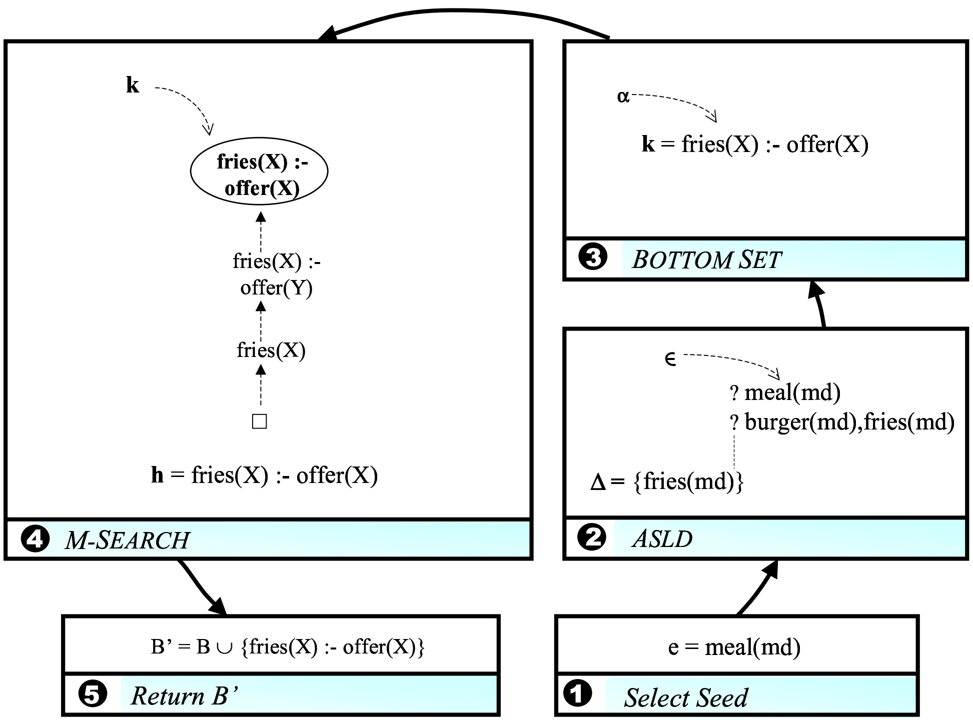

Figure 5.4: Fast Food Example - Solved by HAIL (but not by Progol5)

Background Knowledge
$B = \left\{ \; sad(X) \text{ :- } tired(X), poor(X) \; \right\} \cup$ $\left\{ \begin{array}{l} academic(oli) \\ academic(ale) \\ academic(kb) \end{array} \right\} \cup \left\{ \begin{array}{l} student(oli) \\ lecturer(ale) \\ lecturer(kb) \end{array} \right\}$

Positive Examples	**Negative Examples**
$E^{+} = \left\{ \begin{array}{l} sad(ale) \\ sad(kb) \end{array} \right\}$	$E^{-} = \left\{ \begin{array}{l} \text{:- } sad(oli) \\ \text{:- } poor(oli) \end{array} \right\}$

Head-Declarations	**Body-Declarations**
$M^{+} = \left\{ \begin{array}{l} \mathsf{modeh}[tired(+academic)] \\ \mathsf{modeh}[poor(+academic)] \end{array} \right\}$	$M^{-} = \left\{ \begin{array}{l} \mathsf{modeb}[lecturer(+academic)] \\ \mathsf{modeb}[academic(+academic)] \end{array} \right\}$

Hypothesis
$H = \left\{ \begin{array}{l} tired(X) \\ poor(X) \text{ :- } lecturer(X) \end{array} \right\}$

Figure 5.5: Academic Example

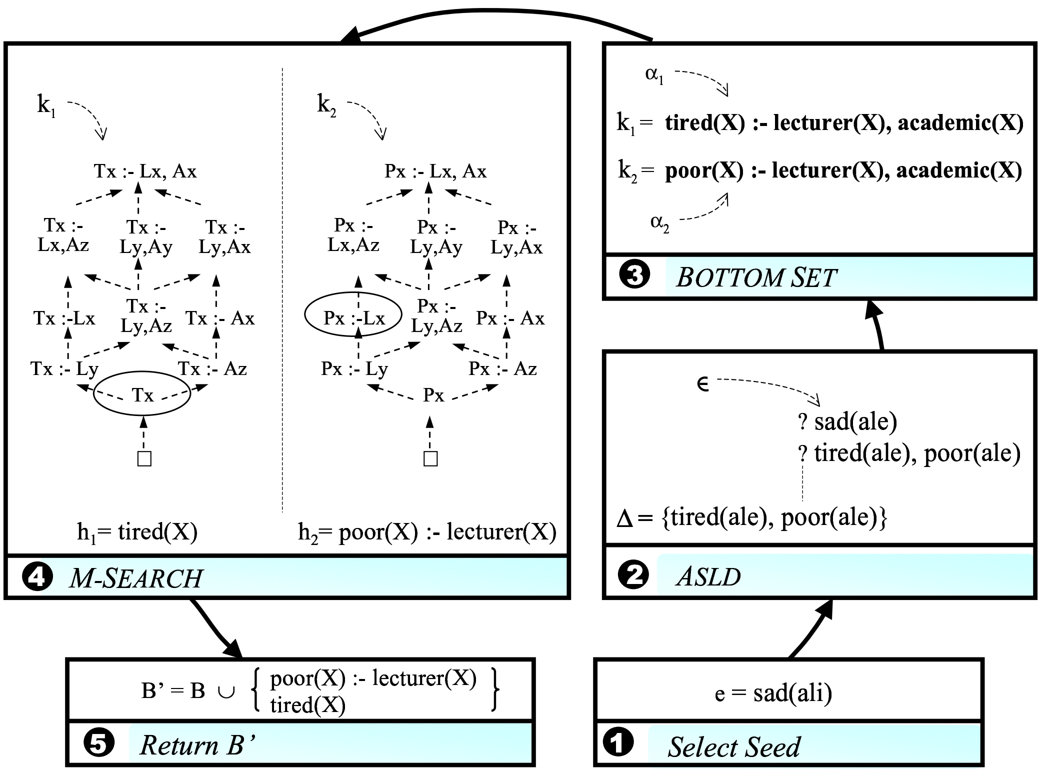

Figure 5.6: Academic Example - Solved by HAIL (but not by Bottom Generalisation)

5.3.4 Redundancy Checking

This section concludes by describing a simple modification of the M-SEARCH routine that enables more compressive generalisations to be computed.

Recall from above that at any given state in the search process, exactly one clause is being considered from each sub-lattice. It may happen, however, that a particular sub-lattice k contributes a clause C, which is already entailed by clauses contributed by the other sub-lattices (relative to the background knowledge). Because the clause C is logically redundant, it should removed, in order not to adversely influence the compression score of the current hypothesis. One way of modelling this situation is by viewing the sub-lattice k as contributing the null-clause: which in this context means no clause at all. The HAIL algorithm is easily modified by replacing the assignment $hs = [head(\mathcal{K})]$ with the new assignment $hs = [\blacksquare, head(\mathcal{K})]$. This means that the algorithm will always begin by checking whether the null clause is an acceptable solution. Note that the null-clause is treated essentially as the empty set, and is therefore defined as having a complexity of zero. This modification enables HAIL to solve the context used previously in Proposition 5.1.4, where the Kernel Set contains two clauses but the hypothesis contains only one.

Example 5.3.7. This example is a meta-level learning problem about object-level learning.

$$\begin{array}{rcl}
B & = & \left\{ \begin{array}{l} groundLiteral(\text{“}p\text{”}) \\ groundLiteral(\text{“}\neg p\text{”}) \\ tautology(bot(B,E))\text{:-}\, in(bot(B,E),\text{“}p\text{”}), in(bot(B,E),\text{“}\neg p\text{”}) \end{array} \right\} \\
E^+ & = & \{\ tautology(bot(B,E))\text{:-}\, entails(B,E)\ \} \\
E^- & = & \{\ \text{:-}\, in(bot(B,E),\text{“}\bot\text{”})\ \} \\
H & = & \{\ in(bot(B,E),X)\text{:-}\, entails(B,E), groundLiteral(X)\ \}
\end{array}$$

Let the meta-level atom $groundLiteral(\text{“}p\text{”})$ represent the fact that the proposition symbol “p” is a ground literal of the object language. Note that by the definition of a literal, its negation “$\neg p$” is also a ground literal of the object language, and so $groundLiteral(\text{“}\neg p\text{”})$.

Let the meta-level term $bot(B,E)$ represents the Bottom Set of the object-level formulae represented by B and E. Note that following standard Prolog convention, the functor *bot* begins with a lower case letter and the variables B and E are upper case letters. Let the meta-level atom $in(bot(B,E),\text{“}p\text{”})$ denote the fact that the object-level atom “p” is contained in the Bottom Set of the object-level formulae represented by B and E. Let the meta-level atom $tautology(bot(B,E))$ denote the fact that the Bottom Set represented by $bot(B,E)$ is a tautology. Note that by the definition of Bottom Set, whenever both $in(bot(B,E),\text{“}p\text{”})$ and $in(bot(B,E),\text{“}\neg p\text{”})$ then it follows $tautology(bot(B,E))$. Represent this fact with the clause $tautology(bot(B,E))\text{:-}\, in(bot(B,E),\text{“}p\text{”}), in(bot(B,E),\text{“}\neg p\text{”})$.

Let the meta-level atom $entails(B,E)$ denote the fact that the object-level formula B entails the object-level formula E. Suppose one wished to explain why the Bottom Set of B and E is a tautology whenever B entails E. Represent this with the positive example $tautology(bot(B,E))\text{:-}\, entails(B,E)$ Let “$\bot$” denote the logical constant for falsity in the object language. Note that “$\bot$” can appear in no Bottom Set. This is because an atom is usually defined as a proposition or predicate symbol, from the signature of a language, applied to an appropriate tuple of terms. But the logical constants are not usually included in the signature of a language, as their interpretation is fixed. Therefore, although they are logical formulae, the logical constants are not, strictly speaking, logical atoms or literals. But the Bottom Set is a set of literals. Therefore, "$\perp$" can appear in no Bottom Set. Represent this with the integrity constraint $\text{:-}\, in(bot(B,E), \text{"}\perp\text{"})$

Given this learning problem (which is summarised above) together with a set of suitable mode declaration, the HAIL procedure can learn the hypothesis containing the single clause $in(bot(B,E),X) \,\text{:-}\, entails(B,E), groundLiteral(X)$ – but only if redundancy checking is enabled. This hypothesis happens to represent the true fact that if B entails E then all ground literals are contained in the Bottom Set of B and e. Details are left to the reader.

Chapter 6

Related Work

The importance of abductive inference in the context of Bottom Generalisation was first realised in [MB00] and [Yam00]. In [MB00], Muggleton and Bryant suggest that Progol5 can be seen as a procedure for efficiently generalising the atoms computed by the STARTSET routine, which they view as implementing a form of abduction based on contrapositive reasoning. This report confirms the view of Muggleton and Bryant by showing that STARTSET performs abduction from *standardised* inputs, but reveals that STARTSET is *incomplete* with respect to Bottom Generalisation. In [Yam00] it is shown that given definite clauses B and e, then $Bot^{-}(B,e)$ is the set of atoms in the least Herbrand model of the definite theory consisting of B and the Skolemised body of e, and $Bot^{+}(B,e)$ is the set of atoms abducible by SOLDR-resolution from this program given as goal the Skolemised head of e. The Kernel semantics presented in this report can be seen as a generalisation of these results that exploits multiple atom abductive hypotheses. In [Yam00], Yamamoto describes a procedure that incorporates explicit abduction within Bottom Generalisation. Atoms in the head and body of the Bottom Set are computed by separate abductive and deductive procedures, and hypotheses are formed by generalising the computed atoms. However, this procedure is non-deterministic, and is restricted to definite clause logic. Yamamoto shows that his procedure is able to induce a single clause or a set of facts for each seed example, but he conjectures that it would be difficult to extend the procedure to induce conjunctions of definite clauses. The proof procedure and semantics described in this report can be seen as generalising those in [MB00] and [Yam00] by constructing Horn theories not derivable by Bottom Generalisation. But still, not all hypotheses can be found with this new approach, as can be seen using the following example due to Yamamoto [Yam97]. If $B = \{even(0)\} \cup \{even(s(X)) \coloneq odd(X)\}$ and $e = odd(s(s(s(0))))$, then the hypothesis $h = odd(s(X)) \coloneq even(X)$ is not derivable by Kernel Set Subsumption, or by Kernel Generalisation, as $\mathcal{K}er(B,e) = \{odd(s(s(s(0)))) \coloneq even(0)\}$ and $h \not\models \mathcal{K}er(B,e)$. Note that in this example $\mathcal{K}er(B,e) = Bot(B,e)$.

Complete methods of hypothesis finding for full clausal logic are proposed in [YF00] and [Ino01a]. In [YF00], Yamamoto and Fronhöfer describe a technique based on *Residue Hypotheses*. Very briefly, the *Residue* of a ground theory G, written $Res(G)$, is the ground theory consisting of all non-tautological clauses that contain the negation of one literal from each clause in G. A *Residue Hypothesis* of two clausal theories B and E, is defined as the Residue of a subset of the ground instances of clauses in B and clauses in the Residue of the Skolemisation of E. A hypothesis H is derived by the Residue Procedure from B and E iff H generalises a Residue Hypothesis of B and E. If the example consists of a single clause e, then a theory H is derived by the Residue Procedure from B and e iff $H \models Res(Gnd(B_e))$ where $Gnd(B_e)$ denotes the ground instances of the complementary extension $B_e = B \cup \overline{e}$. Compare this with Kernel Set Subsumption, which derives a theory H iff $H \sqsupseteq \mathcal{K}$ where $\mathcal{K}$ is a kernel set of B and e. Both procedures derive hypotheses by generalising a ground theory constructed from B and e. For example, if $B = \{p \text{:-} q(a), q(b)\}$ and $e = p$ then $H = \{q(X)\}$ is derived by the Residue Procedure with $Res(Gnd(B_e)) = \{q(a), p\} \cup \{q(b), p\}$, and is derivable by Kernel Set Subsumption with $\mathcal{K} = \{q(a)\} \cup \{q(b)\}$, but not by Bottom Generalisation, as shown in Proposition 5.1.4. In [Ino01a], Inoue describes a technique called *Consequence Finding Induction* or *CF-Induction*, which is based on the concepts of *Production Fields* and *Characteristic Clauses*. Very briefly, a Production Field defines a syntactic language bias on the hypothesis space, and a Characteristic Clause of two clausal theories B and E, is a non-tautological clause entailed $B \wedge \overline{E}$ that is expressed in the language of some Production Field P, and is not properly subsumed by any other such clause. A hypothesis H is derived by CF-Induction iff H generalises the complement of a theory $CC(B, E)$ containing a set of Characteristic Clauses. For the example above, $H = \{q(X)\}$ is derived by CF-Induction with $CC = \{p \text{:-} q(a), q(b)\} \cup \{\text{:-} p\}$ since $\overline{CC}$ is equivalent to the theory $\{q(a), p\} \cup \{q(b), p\}$, and $q(X) \models \overline{CC}$. But because the Residue Procedure and CF-Induction are more general than HAIL, they must search a correspondingly larger hypothesis space, which makes them in general highly nondeterministic and computationally expensive. It is believed, however, that by building directly on the success of Progol, practical ILP systems can be developed for HAIL, that search a progressively larger hypothesis space.

Chapter 7

Conclusions and Future Work

This chapter concludes with a summary and discussion of future work. First, Section 7.1 summarises the research so far. Section 7.2 suggests some possible extensions of the research.

7.1 Summary

This report began with a comprehensive formalisation of the ILP task, and by defending this formalisation against some commonly encountered alternatives. Particular concerns were raised over the standard formulations of the consistency criterion, which were shown to give counter-intuitive behaviour in full clausal logic. The report continued with an analysis of the ILP system Progol5 and the underlying inference method of Bottom Generalisation. The relationship between Bottom Generalisation and relative subsumption was investigated revealing a more intuitive and more general connection than was previously realised. The Progol5 STARTSET and standardisation procedures were investigated and both were found to be incomplete with respect to Bottom Generalisation. The notions of *vacuous literals* and *c*-derivations* were introduced to better characterise these incompletenesses. It was argued that while the former incompleteness may in fact be beneficial, the latter comprises a highly undesirable limitation of the Progol5 proof procedure.

This report went on to propose the approach of *Hybrid Abductive Inductive Learning* in order to overcome this newly discovered incompleteness of Progol5, in addition to some of the inherent semantical limitations of Bottom Generalisation. In particular, the semantics of Kernel Generalisation was developed and shown extend that of Bottom Generalisation. A refinement of this semantics, called Kernel Set Subsumption, was specially adapted for Horn clause logic. The HAIL proof procedure was introduced and shown to generalise Progol5 by overcoming the newly identified incompleteness and by learning multiple clauses in response to a single seed example. This procedure has been designed to operate on precisely the same inputs as Progol5, and to exploit as much existing Progol5 technology as possible, but to overcome several procedural and semantic limitations believed to be of immediate practical significance. These benefits were realised by combining deductive, abductive and inductive reasoning within a learning cycle that exploits multiple-atom abductive hypotheses. Finally, the notion of *K-derivation* was introduced to better characterise the hypotheses derivable by HAIL.

7.2 Continuation

Two things are necessary in order to complete this research. First, on the practical side, a prototype implementation of the HAIL proof procedure should be produced and applied to a small case study. One possibility would be to generalise the existing Progol5 implementation of Muggleton [MB00] and for the abductive phase to use an existing ASLD implementation, such as the A-System of Van Nuffelen, Kakas and Denecker [KND01]. The case study could consist of an existing bioinformatic data set previously used by Progol. This would be in the correct format for HAIL, and would enable some form of comparison between the Progol and HAIL approaches.

Second, on the theoretical side, precise soundness completeness results are still required for some of the semantics and proof procedures presented in this report. All of the procedures must be specified mathematically, and their correctness should be proved. This will require accounting for basicality and local minimality of ASLD hypotheses, stable model semantics, and hypothesis space constraints and compression. Completeness results should be obtained for some suitable problem class. A likely candidate is constrained acyclic programs – as strong completeness results can be obtained for the ASLD procedure under these conditions, and yet this setting is sufficient to illustrate the advantages of HAIL over Progol5.

An obvious extension of the HAIL approach is to consider the possibility of enlarging the class of derivable hypotheses by somehow interleaving the abductive, deductive and inductive phases of the HAIL learning cycle. In addition, the use of negation as failure and of constructive abduction could be investigated. Also interesting, would be to explore various control strategies to restrict or prioritise the hypotheses returned by the abductive phase, and to analyse different heuristics for increasing the efficiency of the search procedure.

Acknowledgements

I would like to thank my supervisors Alessandra Russo and Krysia Broda for their valuable suggestions and constructive criticism of numerous drafts of this report. I would like to thank Steve Muggleton for our detailed discussions concerning Progol4 and Progol5. I am grateful to Francesca Toni and Tony Kakas for their helpful advice.

Appendix

Lemma 7.2.1 (Insertion of Substitutions). *Given two clauses C and D, and any substitution θ. Then*

$$C \subseteq D \textit{ implies } C\theta \subseteq D\theta$$

Lemma 7.2.2 (Removal of Substitutions). *Given two clauses C and D, and a substitution $\sigma = \{Y_1/c_1, \ldots, Y_m/c_m\}$ binding variables to distinct constants not appearing in C or D. Then*

$$C\sigma \subseteq D\sigma \textit{ implies } C \subseteq D$$

Lemma 7.2.3 (Factorisation of Substitutions). *Given a formula F, and a substitution $\theta = \{X_1/t_1, \ldots, X_n/t_n\}$ binding variables to ground terms, and a substitution $\sigma = \{Y_1/c_1, \ldots, Y_m/c_m\}$ binding variables to distinct constants.*

$$\textit{let } \phi = \{X/t' \mid X/t \in \theta \textit{ and } t' = t\sigma^{-1}\}$$

Then the following statement is true:

$$\textit{if } F\theta \textit{ is ground then} \quad F\theta = F\phi\sigma$$

Proof. The composition $\phi\sigma$ may in general be written $\{X/t'\sigma \mid X/t' \in \phi\} \cup \{Y/c \mid Y/c \in \sigma$ and $Y/t \notin \phi$ for any term $t\}$. For $F\theta$ to be ground, every variable in F must bound by θ, and must therefore also be bound by ϕ, and consequently cannot occur in the second set, which can have no effect on F. Since each t is a ground term, and σ^{-1} replaces distinct terms with distinct variables, it follows that $t'\sigma = t$. And therefore $\phi\sigma$ binds each variable in F to the same ground term as θ. Hence $f\theta = F\phi\sigma$ ▮

Bibliography

[ABW87] K. Apt, H. Blair, and A. Walker. Foundations of deductive databases and logic programming. In J. Minker, editor, *Towards a Theory of Declarative Knowledge*, pages 89–148. Morgan Kaufmann, 1987.

[CL73] C. Chang and R. C. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.

[Cla78] K. L. Clark. *Logic and Data Bases*, chapter Negation as failure rule, pages 293–322. Plenum Press, first edition, 1978.

[CP86] P.T. Cox and T. Pietrzykowski. Causes for events: Their computation and application. In J.H. Siekmann, editor, *8th International Conference on Automated Deduction*, volume 230 of *Lecture Notes in Computer Science*, pages 608–621. Springer, 1986.

[End72] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, Inc, 1972.

[FK00] P.A. Flach and A.C. Kakas, editors. *Abduction and Induction: essays on their relation and integration*, volume 18 of *Applied Logic Series.* Kluwer Academic Publishers, first edition, 2000.

[FMPS98] P. Finn, S. Muggleton, D. Page, and A. Srinivasan. Pharmacophore discovery using the inductive logic programming system Progol. *Journal of Machine Learning*, 30:241–271, 1998.

[GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R.A. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming*, pages 1070–1080. The MIT Press, 1988.

[GRS91] A. Van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[Ino01a] K. Inoue. Induction, abduction, and consequence-finding. In C. Rouveirol and M. Sebag, editors, *Proceedings 11th International Conference on Inductive Logic Programming*, volume 2157 of *Lecture Notes in AI*, pages 65–79. Springer Verlag, 2001.

[Ino01b] K. Inoue. Inverse entailment for full clausal theories. In *LICS-2001 Workshop on Logic and Learning*, 2001.

[KD02] A.C. Kakas and M. Denecker. Abduction in logic programming. In A.C. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, volume 2407 of *Lecture Notes in Computer Science*, pages 402–436. Springer, 2002.

[KKT92] A.C. Kakas, R.A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1992.

[KM90] A.C. Kakas and P. Mancarella. Database updates through abduction. In *16th International Conference on Very Large Databases (VLDB)*, pages 650–661. Morgan Kaufmann, 1990.

[KND01] A. Kakas, B. Van Nuffelen, and M. Denecker. A-system : Problem solving through abduction. In *Proceedings of IJCAI'01 - Seventeenth International Joint Conference on Artificial Intelligence*, pages 591–596, 2001.

[Llo87] J. W. Lloyd. *Foundations of Logic Programming.* Springer Verlag, 1987.

[MB00] S.H. Muggleton and C.H. Bryant. Theory completion using inverse entailment. *Lecture Notes in Computer Science*, 1866:130–146, 2000.

[MF01] S.H. Muggleton and J. Firth. Cprogol4.4: A tutorial introduction. In S. Dzeroski and N. Lavrac, editors, *Relational Data Mining*, pages 160–188. Springer-Verlag, 2001.

[Mic84] R. S. Michalski. A theory and methodology of inductive learning. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an Artificial Intelligence Approach.* Springer-Verlag, 1984.

[MR94] S.H. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

[Mug91] S.H. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[Mug95] S.H. Muggleton. Inverse entailment and progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[NCdW97] S.H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence.* Springer, first edition, 1997.

[Plo71] G.D. Plotkin. *Automatic Methods of Inductive Inference.* PhD thesis, Edinburgh University, 1971.

[Prz89] T. C. Przymusinski. On the declarative and procedural semantics of logic programs. *Journal of Automated Reasoning*, 5:167–205, 1989.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

[SMSK96] A. Srinivasan, S.H. Muggleton, M.J.E. Sternberg, and R.D. King. Theories for mutagenicity: A study in first-order and feature-based induction. *Journal of Artificial Intelligence*, 85(1-2):277–299, 1996.

[Sti88] M.E. Stickel. A Prolog technology theorem prover: Implementation by an extended Prolog compiler. In J. H. Siekmann, editor, *Journal of Automated Reasoning*, volume 4(4), pages 353–380, 1988.

[TMS01] M. Turcotte, S. H. Muggleton, and M. J. E. Sternberg. Automated discovery of structural signatures of protein fold and function. *Journal of Molecular Biology*, 306:591–605, 2001.

[Yam96] A. Yamamoto. Improving theories for inductive logic programming systems with ground reduced programs. Technical Report AIDA-96-19, Fachgebiet Intellektik, Fachbereich Informatik, Technische Hochschule Darmstadt, 1996.

[Yam97] A. Yamamoto. Which hypotheses can be found with inverse entailment? In S. Džeroski and N. Lavrač, editors, *Proceedings of the 7th International Workshop on Inductive Logic Programming*, volume 1297, pages 296–308. Springer-Verlag, 1997.

[Yam99] A. Yamamoto. An inference method for the complete inverse of relative subsumption. *New Generation Computing*, 17(1):99–117, 1999.

[Yam00] A. Yamamoto. *Abduction and Induction: essays on their relation and integration*, chapter Using Abduction for Induction based on Bottom Generalisation. Volume 18 of Flach and Kakas [FK00], first edition, 2000.

[YF00] A. Yamamoto and B. Fronhöfer. Hypothesis finding via residue hypotheses with the resolution principle. In *Proceedings of the 11th International Conference on Algorithmic Learning Theory*, volume 1968 of *Lecture Notes in Computer Science*, pages 156–165. Springer-Verlag, 2000.