DOV GABBAY AND NICOLA OLIVETTI

# LECTURE NOTES ON ALGORITHMIC PROOF-THEORY

EEF SCHOOL ON
FOUNDATIONS OF DEDUCTION AND THEOREM PROVING

HERIOT-WATT UNIVERSITY, EDIMBURGH
6–16 APRIL 2000

# CONTENTS

# CHAPTER 1

# INTRODUCTION

The landscape of non-classical logics applications in computer science and artificial intelligence is wide and varied. Modal and temporal logics have been profitably applied to verification and specification of concurrent systems [Manna and Pnueli, 1981; Pnueli, 1981]. In the area of artificial intelligence as well as distributed systems, the problem of reasoning about knowledge, belief and action has received much attention; modal logics [Turner, 1985; Halpern and Moses, 1990] have been seen to provide the formal language to represent this type of reasoning. Relevance logics have been applied in natural language understanding and database updating [Martins and Shapiro, 1988]. Lambek's logic [1958] and its extensions are currently used for natural language processing. Another source of interest in non-classical logics, and in particular in the so-called substructural logics (with the prominent case of *linear logic* [Girard, 1987]) has originated from the functional interpretation provided by the Curry–Howard's isomorphism between formulas and types in functional languages (see [Hindley, 1997] for an introduction and references therein, see also also[Wansing, 1990; Gabbay and de Queiroz, 1992]).

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

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

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

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

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

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

1. given a formula $A$ we want to know whether $\vdash A$, that is whether $A$ is a theorem of $\mathcal{L}$;
2. we are given a set $\Gamma$ containing 10,000 formulas and a formula $A$ and we want to know whether $\Gamma \vdash A$;
3. we are given a set of formulas $\Gamma$ and we are asked to generate all atomic propositions which are entailed by $\Gamma$;
4. We are given a formula $A$ and a set of formulas $\Gamma$ such that $\Gamma \not\vdash A$. We are asked to find a set of atomic propositions $S$ such that $\Gamma \cup S \vdash A$.

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

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

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

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

The goal-directed paradigm we follow in this work is particularly well-suited to deduction which may involve a great amount of data (that is what we have qualified as deductive database perspective). Moreover goal-directed deductive procedures can be used to design abductive reasoning procedures [Eshghi, 1989], although this point will not be developed further in the present work.

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

$$\Delta \vdash^{?} A$$

the query 'does $A$ follows from $\Delta$?' (in a given logic). The deduction is *goal-directed* in the sense that the next step in a proof is determined by the form of the current goal: the goal is stepwise decomposed, according to its logical structure, until we reach its atomic constituents. An atomic goal $q$ is then matched with the 'head' of a formula $G' \rightarrow q$ (if any, otherwise we fail) in the database, and its 'body' $G'$ is asked in turn. This is what happens with the logic programming approach to Horn-clause computation in intuitionistic or classical logic, namely a Horn-clause can be read as a procedure

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

reduces to

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

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

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

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

## NOTATION AND BASIC NOTIONS

### Formulas

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

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

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

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

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

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

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

**Implicational formulas**

In much of the work we will be concerned with *implicational formulas*. These formulas are generated from a set of atoms by the only connective $\rightarrow$. We adopt some specific notations for them. We sometimes distinguish the *head* and the *body* of an implicational formula[1]. The head of a formula $A$ is its rightmost nested atom, whereas the body is the *list* of the antecedents of its head. Given a formula $A$, we define $Head(A)$ and $Body(A)$ as follows:

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

$Body(q) = (\ )$, if $q$ is an atom,
$Body(A \rightarrow B) = (A) * Body(B)$,

where $(A) * Body(B)$ denotes the list beginning with $A$ followed by $Body(B)$.

Dealing with implicational formulas, we assume that implication associates on the right, i.e. we write

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

instead of

$$A_1 \rightarrow (A_2 \rightarrow \ldots \rightarrow (A_{n-1} \rightarrow A_n) \ldots).$$

It turns out that every formula $A$ can be written as

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

where we obviously have.

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

**Harrop and Horn formulas**

Two special classes of formulas will be of interest in this work, namely Harrop and Horn formulas. To introduce the former, let us define the two types of formulas DH and GH by mutual induction as follows:

$$DH := q \mid GH \rightarrow DH \mid DH \wedge DH$$
$$GH := q \mid GH \wedge GH \mid GH \vee GH \mid DH \rightarrow GH.$$

A formula is *Harrop* if it is defined according to the DH-clauses above.

REMARK 1.1. *Harrop formulas* are essentially propositional formulas that do not contain *positive* occurrences of disjunction. When considered, DH-formulas will be allowed as constituents of the database, whereas GH formulas will be allowed to be asked as goals.

To introduce *Horn* formulas, let us define similarly HO and GO formulas:

[1] This terminology is reminiscent of logic programming [Lloyd, 1984].

$$HO := \ q \mid GO \to HO \mid HO \wedge HO$$
$$GO := \ q \mid GO \wedge GO \mid GO \vee GO.$$

A formula is *Horn* if it is defined according to the HO-clauses above.

REMARK 1.2. Horn formulas are hence Harrop formulas without nested implications. In any logical system where the following equivalences hold (for instance in Classical and Intuitionistic logic):

$$A \to (B \to C) \quad \equiv \quad (A \wedge B) \to C \tag{1.1}$$
$$(A \vee B) \to C \quad \equiv \quad (A \to C) \wedge (B \to C) \tag{1.2}$$
$$A \to (B \wedge C) \quad \equiv \quad (A \to B) \wedge (A \to C) \tag{1.3}$$

we have that any Horn formula can be rewritten as a conjunction of implicational formulas of the form

$$p_1 \to p_2 \to \ldots \to p_n \to q,$$

where all $p_i$ and $q$ are atoms. Similarly, in any logical system where 1.1 holds any Harrop formula is equivalent to a *conjunction* of DH-formulas defined only by

$$DH := \ q \mid GH \to DH.$$

**Multisets**

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

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

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

The *support* of a multiset $\alpha$, denoted by $\bar{\alpha}$ is the set of $x \in S$, such that $\alpha(x) > 0$.

In order to display the elements of a multiset $\alpha$, we use the notation

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

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

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

(Strong containment) We define the relation $\alpha \subseteq\mid \beta$ as follows

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

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

CHAPTER 2

# INTUITIONISTIC AND CLASSICAL LOGICS

## 1 INTUITIONISTIC LOGIC

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

(1) Implication group:

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

(2) Conjunction group:

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

(3) Disjunction group:

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

(4) Falsity:

1. $\perp \to A$,

(5) Negation group:

1. $(A \to \neg B) \to B \to \neg A$,

2. $\neg A \to A \to B$.

In addition, it contains the *Modus Ponens* rule:

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

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

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

$$\neg A \to (A \to \bot) \text{ and } (A \to \bot) \to \neg A$$

are theorems of the above axiom system. The reader is invited to check them by herself.If add to **I** any of the axioms below we get classical logic **C**:

**Alternative axioms for classical logic**

(Peirce's axiom) $((A \to B) \to A) \to A$

(double negation) $\neg\neg A \to A$,

(excluded middle) $\neg A \vee A$,

($\vee, \to$-distribution) $[(A \to (B \vee C)] \to [(A \to B) \vee C]$.

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

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

DEFINITION 2.1. Given a propositional language L, a Kripke model for L is a structure of the form $M = (W, \leq, w_0, V)$, where $W$ is a non-empty set, $\leq$ is a reflexive and transitive relation on $W$, $w_0 \in W$, $V$ is a function of type: $W \longrightarrow Pow(Var_{\mathcal{L}})$, that is $V$ maps each element of $W$ to a set of propositional variables of L. We assume the following conditions:

(1) $w_0 \leq w$, for all $w \in W$;

(2) $w \leq w'$ implies $V(w) \subseteq V(w')$;

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

Given $M = (W, \leq, w_0, V)$, $w \in W$, for any formula $A$ of L, we define '$A$ is true at $w$ in $M$', denoted by $M, w \models A$ by the following clauses:

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

We say that $A$ is *valid* in $M$ if $M, w_0 \models A$ and we denote this by $M \models A$. We say that $A$ is *valid* if it is valid in every Kripke model $M$. We also define a notion of entailment between sets of formulas and formulas. Let $\Delta = \{A_1, \ldots, A_n\}$ be a set of formulas and $B$ be a formula, we say that $\Gamma$ *entails* $B$ denoted by $\Gamma \models B$ [1] iff

$$M, w_0 \models A_i \text{ for all } A_i \in \Gamma, \text{ then } M, w_0 \models B.$$

One can think of a Kripke model $M = (W, \leq, w_0, V)$ as a tree with root $w_0$. The definition of entailment is by no means restricted to finite $\Gamma$s. Whenever $\Gamma$ is finite, $\Gamma \models A$ holds iff $\bigwedge \Gamma \to A$ is valid. It is easy to prove that condition (2) implies that in every model $M = (W, \leq, w_0, V)$, for any $w, w' \in W$, if $w \leq w'$ and $M, w \models A$, then $M, w \models A$. By this property, we immediately obtain:

$$M, w_0 \models A \text{ iff for all } w \in W\ M, w \models A.$$

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

A standard argument shows that the propositional calculus given above is sound and complete with respect to Kripke models. For this and the other background facts mentioned in this section, the reader is referred to any of the texts quoted at the beginning of the chapter.

THEOREM 2.2. *For any formula $A$, $A$ is a theorem of* **I** *iff $A$is valid.*

This completeness theorem can be sharpened to *finite* Kripke models (finite trees) that is, models $M = (W, \leq, w_0, V)$, where $W$ is a finite set.

THEOREM 2.3. *For any formula $A$, $A$ is a theorem of* **I** *iff $A$ is valid in every* finite *Kripke model.*

[1] To be precise, we should write $\models_{\mathbf{I}}$ (and $\vdash_{\mathbf{I}}$) to denote validity and entailment (respectively, provability and logical consequence) in intuitionistic logic **I**. To avoid burdening the notation, we usually omit the subscript if there is no risk of confusion.

Classical interpretations can be thought as degenerated Kripke models $M = (W, \leq, w_0, V)$, where $W = \{w_0\}$.

We give a third presentation of intuitionistic logic in terms of consequence relation. Let $\Gamma$ be a set of formulas and $A$ be a formula. We write

$$\Gamma \vdash A$$

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

DEFINITION 2.4 (Consequence relation for intuitionistic logic). Let $\Gamma \vdash A$ be defined as the smallest relation which satisfies:

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

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

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

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

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

THEOREM 2.5. *Let $\vdash$ be the smallest consequence relation satisfying (1)–(4) above. Then $\Gamma \vdash A$ holds iff $\Gamma \models A$.*

Classical logic can be obtained as the smallest consequence relation satisfying (1)–(4) together with an additional condition corresponding to an axiom for classical logic as given above, for instance:

$$\frac{\Delta, A \vdash B \vee C}{\Delta \vdash (A \to B) \vee C}$$

or

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

## 2 RULES FOR INTUITIONISTIC IMPLICATION

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

$$\Delta \ \vdash^? \ A.$$

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

DEFINITION 2.6.

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

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

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

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

  We say that $C$ is used in this step.

A *derivation* $\mathcal{D}$ of a query $Q$ is a tree whose nodes are queries. The root of $\mathcal{D}$ is $Q$, and the successors of every non-leaf query are determined by exactly one applicable rule (*implication or reduction*) as described above.

We say that $\mathcal{D}$ is *successful* if the *success rule* may be applied to every leaf of $\mathcal{D}$.

We finally say that a query $Q$ *succeeds* if there is a successful derivation of $Q$.

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

A last observation: the success of a query is defined in a non-deterministic way; a query succeeds if there is a successful derivation. To transform the proof rules into a deterministic algorithm one should give a method to search a successful derivation tree. In this respect we agree that when we come to an atomic goal we first try to apply the success rule and if it fails we try the reduction rule. Then the only choice we have is to indicate which formula among those of the database whose head matches the current atomic goal, we use to perform a reduction step, if there are more than one. Thinking of the database as a list of formulas, we can choose the first one and remember the point up to which we have scanned the database as a backtracking point. This is exactly as in conventional logic programming [Lloyd, 1984].

EXAMPLE 2.7. We check

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

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

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

PROPOSITION 2.8.

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

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

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

**Proof.**

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

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

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

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

$$
\begin{array}{ccc}
 & (1)\ \Gamma \vdash^? b & \\
 & | & \\
 & (2)\ \Gamma \vdash^? p & \\
 & | & \\
 & (3)\ \Gamma \vdash^? a & \\
 \swarrow & & \searrow \\
(4)\ \Gamma \vdash^? a \to b & & (8)\ \Gamma \vdash^? c \\
| & & | \\
(5)\ \Gamma, a \vdash^? b & & (9)\ \Gamma \vdash^? p \to d \\
| & & | \\
(6)\ \Gamma, a \vdash^? p & & (10)\ \Gamma, p \vdash^? d \\
| & & | \\
(7)\ \Gamma, a \vdash^? a & & (11)\ \Gamma, p \vdash^? b \\
 & & | \\
 & & (12)\ \Gamma, p \vdash^? p
\end{array}
$$

Figure 2.1. Derivation for Example 2.7.

By the induction hypothesis, each of the above queries succeeds.

(b) is proved by induction on the height of a successful computation.

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

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

Since we are dealing with the implicational fragment, by Theorem 2.3, it is enough to show that the consequence relation defined by:

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

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

THEOREM 2.9. *If* $\Delta \vdash_p A$ *then* $\Delta \vdash A$.

**Proof.** By induction on the height $h$ of a successful derivation of $\Delta \vdash^? A$. Let $h = 0$. If $\Delta \vdash^? A$ succeeds by a derivation of height $0$, then $A$ is an atom and

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

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

By the induction hypothesis, for $i = 1, \ldots, n$,

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

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

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

so that, by deduction theorem also

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

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

In order to prove completeness it is sufficient to show that $\vdash_p$ satisfies the conditions of intuitionistic consequence relation, namely identity, monotony, deduction theorem and cut. That the first three properties are satisfied is proved in pProposition 2.8. So, the only property we have to check is cut. We prove closure under cut in the next theorem.

THEOREM 2.10 (Admissibility of Cut). *If*

$\Delta, A \vdash_p B$ *and* $\Gamma \vdash_p A$ *imply* $\Delta, \Gamma \vdash_p B$.

**Proof.** Assume (1) $\Delta, A \vdash_p B$ and (2) $\Gamma \vdash_p A$.

The theorem is proved by induction on lexicographically-ordered pairs $(c, h)$, where $c = cp(A)$, and $h$ is the height of a successful derivation of (1), that is of $\Delta, A \vdash^? B$. Suppose first $c = 0$, then $A$ is an atom $p$, and we proceed by induction on $h$. If $h = 0$, $B$ is an atom $q$ and either $q \in \Delta$ or $q = p = A$. In the first case, the claim trivially follows by Proposition 2.8. In the second case it follows by hypothesis (2) and Proposition 2.8.

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

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

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

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

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

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

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

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

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

By hypothesis (2) we can conclude that

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

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

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

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

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

This concludes the proof. ■

THEOREM 2.11 (Completeness for **I**). *If* $\Delta \vdash A$, *then* $\Delta \vdash_p A$.

**Proof.** We know that $\vdash$ is the smallest consequence relation satisfying (identity), (monotony), (deduction theorem) and (cut). By Proposition 2.8 and Theorem 2.10, $\vdash_p$ satisfies these properties as well. Then, the claim follows by the minimality of $\vdash$. ■

## 3 BOUNDED RESOURCE DEDUCTION FOR IMPLICATION

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

Consider the data and query below:

$$q \rightarrow q \ \vdash^? \ q.$$

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

**Historical rule for $\rightarrow$**

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

**Historical Reduction**

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

Thus the computation for our example above becomes:

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

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

One way to perform loop-checking in a more efficient way has been described in [Heudering *et al.*, 1996]. The idea is to avoid repeated insertions of the same formula in the database, taking advantage of the fact that the database is a *set* of formulas. One simply records the history of the atomic goals asked from the same database; the history is cleared when a new formula is inserted in the database. This idea is implemented in a terminating proof system for full intuitionistic propositional logic in the form of a sequent calculus. We reformulate here this variant of loop checking adapted to our context.

**An improved Historical loop-checking**

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

**Rule 1 for $\rightarrow$**
$\Delta \vdash^? A \rightarrow B, H$ succeeds
if $A \notin \Delta$ and $\Delta, A \vdash^? B, \emptyset$ succeeds.

**Rule 2 for $\rightarrow$**
$\Delta \vdash^? A \rightarrow B, H$ succeeds
if $A \in \Delta$ and $\Delta \vdash^? B, H$ succeeds.

**Reduction Rule**
$\Delta \vdash^? q, H$ succeeds,
if $q \notin H$ and for some $C_1 \rightarrow C_2 \rightarrow \ldots C_n \rightarrow q$ in $\Delta$ we have that for all $i$
$\Delta \vdash^? C_i, H * (q)$ succeeds.

EXAMPLE 2.12.

$$\begin{array}{rll}
 & \vdash^? & ((p \rightarrow q) \rightarrow q) \rightarrow (q \rightarrow p) \rightarrow p,\ \emptyset, \\
(p \rightarrow q) \rightarrow q & \vdash^? & (q \rightarrow p) \rightarrow p,\ \emptyset, \\
(p \rightarrow q) \rightarrow q, q \rightarrow p & \vdash^? & p,\ \emptyset, \\
(p \rightarrow q) \rightarrow q, q \rightarrow p & \vdash^? & q,\ (p), \\
(p \rightarrow q) \rightarrow q, q \rightarrow p & \vdash^? & p \rightarrow q,\ (p, q), \\
(p \rightarrow q) \rightarrow q, q \rightarrow p, p & \vdash^? & q,\ \emptyset, \\
(p \rightarrow q) \rightarrow q, q \rightarrow p & \vdash^? & p \rightarrow q,\ (q), \\
(p \rightarrow q) \rightarrow q, q \rightarrow p & \vdash^? & q,\ (q), \\
 & & fail
\end{array}$$

In this way one is able to detect a loop (the same atomic goal repeats from the same database), without having to record each pair (database goal).

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

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

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

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

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

Before we proceed, we need to give a formal definition of these concepts. We give Definition 2.13 below and it should be compared with Definition 2.6 of the previous section.

Since we take care of usage of formulas, it is natural to regard multiple copies of the same formula as distinct. This means that databases can now be considered as *multisets* of formulas. In order to keep the notation simple, we use the same notation as in the previous section. From now on, $\Gamma, \Delta$, etc. will range on multisets of formulas, and we will write $\Gamma, \Delta$ to denote the union multiset of $\Gamma$ and $\Delta$, that is $\Gamma \sqcup \Delta$. To denote a multiset $[A_1, \dots A_n]$, if there is no risk of confusion we will simply write $A_1, \dots A_n$ (see the last section of Chapter 1, for the formal definitions of multiset and relative notions). We present three notions of computation by defining their computation trees. These are:

1. the goal-directed computation for intuitionistic logic;
2. locally linear goal-directed computation (LL- computation);
3. linear goal-directed computation.

DEFINITION 2.13. [Goal-directed computation, locally linear goal-directed computation and linear goal-directed computation] We give the computation rules for a query: $\Gamma \ \vdash^? \ G$, where $\Gamma$ is a multiset of formulas and $G$ is a formula.

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

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

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

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

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

  where the following holds
  1. in the case of intuitionistic computation, $\Delta_i = \Delta$;
  2. in the case of locally linear computation, $\Delta_i = \Delta - [B]$;

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

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

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

Sooner or later we will discover the following examples.

EXAMPLE 2.14.

1. $c \to a, (c \to a) \to c \ \vdash^? \ a$
   The formula $c \to a$ has to be used twice in order for $a$ to succeed. This example can be generalised.

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

   Consider the following query:

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

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

EXAMPLE 2.15. Another example in which a formula must be used locally more than once is the following: the database contains

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

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

EXAMPLE 2.16. Let us do the full LL-computation for

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

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

$$
\begin{array}{ccc}
 & a \to b \to c, a \to b, a \ \vdash^? \ c & \\
 \swarrow & & \searrow \\
a \to b, a \ \vdash^? \ a & & a \to b, a \ \vdash^? \ b \\
 & & | \\
 & & a \ \vdash^? \ a
\end{array}
$$

Figure 2.2. Derivation for Example 2.16.

The example above shows that the locally linear proof system of Definition 2.13 is not the same as the linear proof system. First, we do not require that all assumptions must be used, the condition on the success rule. A more serious difference is that we do our 'counting' of how many times a formula is used separately on each path of the computation and not globally for the entire computation. The counting in the linear case is global, as can be seen by the condition in the reduction rule.

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

In contrast with the intuitionistic consequence relation, we can notice that linear computation does *not* satisfy all the structural properties of Definition 2.4. Monotonicity does not hold. Reflexivity holds in the restricted form $A \vdash A$. The cut rule holds in the form

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

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

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

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

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

Therefore, by cut we should get

$a \ \vdash^? \ (a \to b) \to (a \to b \to c) \to c$ succeeds, which, of course, does not hold. Linear computation as defined in Definition 2.13 corresponds to linear logic implication, [Girard, 1987], in the sense that the procedure of linear computation is sound and complete for the implicational fragment of linear logic. This will be shown in Chapter 5, within the broader context of substructural logics. Linear logic has the implication connective $\multimap$ and $A_1, \ldots, A_n \vdash B$ in linear logic means that $B$ can be proved from $A_i$ using each $A_i$ exactly once. A deduction theorem holds, namely that $A_1, \ldots, A_n \vdash B$ is equivalent to $\vdash A_1 \multimap (A_2 \multimap \ldots (A_n \multimap B) \ldots)$.

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

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

On the other hand, as we have seen

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

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

We recall that a formula $A$ is *Horn* if it is an atom, or has the form $p_1 \to \ldots \to p_n \to q$, where all $p_i$ and $q$ are atoms.

PROPOSITION 2.17. *The locally-linear procedure is complete for Horn formulas.*

**Proof.** One may simply observe that in a derivation involving only Horn formulas the database never changes (except for an *initial* argument if the starting goal is non-atomic). By this fact, we do not need to ask the same goal more than once along the same branch, as any subsequent call of the same goal will not have more chances to succeed than its first call. But this means that we do not need to use a database formula in a reduction step more than once along the same derivation branch. ■

### *3.1 Bounded restart rule for intuitionistic logic*

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

Let us examine more closely why we needed in Example 2.14 the formula $c \to a$ several times. The reason was that from other formulas, we got the query $\vdash^? \ a$ and we wanted to use $c \to a$ to continue to the query $\vdash^? \ c$. Why was $c \to a$ not available? Because $c \to a$ had already been used. In other words, $\vdash^? \ a$ as a query, had already been asked and $c \to a$ was used. This means that the next query after $\vdash^? \ a$ in the history was $\vdash^? \ c$.

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

We can therefore compensate for the reuse of $c \to a$ by allowing ourselves to go back in the history to where $\vdash^? \ a$ was, and allow ourselves to ask all queries that come afterwards. We call this type of move *bounded restart*. Let us see what happens to our Example 2.14

| | | |
|---|---|---|
| $(c \to a) \to c$ | [1] | $\vdash^? \ a$ |
| $c \to a$ | [1] | |
| We use the second formula to get | | |
| $(c \to a) \to c$ | [1] | $\vdash^? \ c, (a)$ |
| $c \to a$ | [0] | |
| we continue | | |
| $(c \to a) \to c$ | [0] | $\vdash^? \ c \to a, (a, c)$ |
| $(c \to a)$ | [0] | |
| we continue | | |
| $(c \to a) \to c$ | [0] | $\vdash^? \ a, (a, c)$ |
| $c \to a$ | [0] | |
| $c$ | [1] | |

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

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

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

success.

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

DEFINITION 2.18. [Locally linear computation with bounded restart] In the computation with bounded restart, the queries have the form $\Delta \vdash^? G, H$, where $\Delta$ is a set of formulas and the history $H$ is a sequence of atomic goals. The rules are as follows:

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

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

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

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

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

THEOREM 2.19 (Soundness of locally linear computation with bounded restart). *For the computation of Definition 2.18 we have : if* $\Delta \vdash^? G, (p_1, \ldots, p_n)$ *succeeds then* $\Delta, G \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash G$ *holds.*

THEOREM 2.20 (Completeness of locally linear computation with bounded restart). *For the computation of Definition 2.18 we have : if* $\Delta, G \to p_n, p_n \to p_{n-1}, \ldots, p_2 \to p_1 \vdash G$ *holds, then* $\Delta \vdash^? G, (p_1, \ldots, p_n)$ *succeeds.*

**A Historical Remark**

The locally linear computation with bounded restart, first presented by Gabbay, [1991], seems strongly connected to the contraction-free sequent calculi for intuitionistic logic which have been proposed by many people: Dyckhoff [1999], Hudelmaier [1990], and Lincoln *et al.*, [1991], in a slightly different context. To see the intuitive connection, let us consider the query:

(1) $\Delta, (A \to p) \to q \vdash^? q, \emptyset$

we can step by reduction to $\Delta \vdash^? A \to p, (q)$ and then to $\Delta, A \vdash^? p, (q)$, which, by the soundness theorem corresponds to

(2) $\Delta, A, p \to q \vdash^? q$.

In all the mentioned calculi (1) can be reduced to (2) by a sharpened left-implication rule (here used backwards). This modified rule is the essential ingredient to obtain a contraction-free sequent calculus for **I**, at least for its implicational fragment. A formal connection with these contraction-free calculi has not been studied yet, although we think it is worth studying. It might turn out that LL-computations correspond to *uniform proofs* (in the sense of Miller *et al.*, [1991]) within these calculi.

## *3.2 Restart Rule for Classical Logic*

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

DEFINITION 2.21 (Restart rule in the LL-computation). If $a \in H$, from $\Delta \vdash^? q, H$ step to $\Delta \vdash^? a, H \cup \{q\}$.

The formal definition of *locally linear computation with restart* is Definition 2.18 with the additional restart rule above in place of the bounded restart rule.

EXAMPLE 2.22.

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

use the formula and throw it out to get:

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

and

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

restart

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

success.

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

If we adopt the basic computation procedure for intuitionistic implication Definition 2.6 rather than the LL- computation, we can restrict the restart rule to always choose the *initial goal* as the goal with which we restart. Thus, we do not need to keep the history, but only the initial goal and the rule becomes more deterministic. On the other hand, the price we pay is that we cannot throw out the formulas of database when they are used.

DEFINITION 2.23 (Simple computation with restart). The queries have the form

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

where $G_0$ is a goal. The computation rules are the same as in the basic computation procedure for intuitionistic implication of Definition 2.6 plus the following rule

(Restart) from

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

step to

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

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

It is natural to wonder what we get if we add the restart rule of Definition 2.21 to the basic procedure for intuitionistic logic of Definition 2.6. In the next proposition we show that given the underlying computation procedure of Definition 2.6, restarting from an arbitrary atomic goal in the history is equivalent to restart from the initial goal. To this purpose, let $\vdash^?_{RI}$ and $\vdash^?_{RA}$ be respectively the deduction procedure of Definition 2.23 and the deduction procedure of Definition 2.6 extended by the restart rule of Definition 2.21.

PROPOSITION 2.24. *For any database $\Delta$ and formula $G$, we have*

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

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

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

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

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

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

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

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

$$\vdots$$
$$(2)\ \Delta_2\ \vdash^?_{RA}\ q_2, H_2$$
$$\vdots$$
$$(3)\ \Delta_3\ \vdash^?_{RA}\ q_3, H_3.$$

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

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

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

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

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

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

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

*Soundness and Completeness of Restart from the Initial Goal*

We show that the proof-procedure obtained by adding the rule of restart from the initial goal to the basic procedure for intuitionistic logic defined in Definition 2.6 is sound and complete with respect to classical provability.

We need the notion of complement of a formula which works as its *negation*.

DEFINITION 2.25. Let $A$ be any formula. The complement of $A$, denoted by $Cop(A)$ is the following set of formulas:

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

The set $Cop(A)$ represents the negation of $A$ in our implicational language which does not contain neither $\neg$, nor $\perp$.

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

LEMMA 2.26. *(1)* $\Gamma\ \vdash^?\ A$ *succeeds by using restart rule iff (2)* $\Gamma \cup Cop(A)\ \vdash^?\ A$ *succeeds without using restart, that is by the intuitionistic procedure of Definition 2.6.*

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

We will prove first that $\Delta \vdash A$ in classical logic iff $\Delta \cup Cop(A) \vdash^? A$ succeeds by the procedure defined in Definition 2.6.

LEMMA 2.27. *For any database $\Delta$ and formulas $G$ such that $\Delta \supseteq Cop(G)$, and for any goal $A$, conditions (a) and (b) below imply condition (c):*

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

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

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

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

THEOREM 2.28. *For any $\Delta$ and $A$, (a) is equivalent to (b) below:*

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

*(b)* $\Delta \cup Cop(A) \vdash^? A$ *succeeds by the intuitionistic procedure defined in Definition 2.6.*

**Proof.**

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

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

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

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

2. Show that (a) implies (b).
We prove that if $\Delta \cup Cop(A) \ \vdash^? \ A$ does not succeed then $\Delta \nvdash A$ in classical logic. Let $\Delta_0 = \Delta \cup Cop(A)$. We define a sequence of databases $\Delta_n, n = 1, 2 \dots$ as follows:
Let $B_1, B_2, B_3, \dots$ be an enumeration of all formulas of the language.
Assume $\Delta_{n-1}$ has been defined and assume that $\Delta_{n-1} \ \vdash^? \ A$ does not succeed. We define $\Delta_n$:
If $\Delta_{n-1} \cup \{B_n\} \ \vdash^? \ A$ does not succeed, let $\Delta_n = \Delta_{n-1} \cup \{B_n\}$. Otherwise from Lemma 2.27 we must have:
$\Delta_{n-1} \cup$ Cop $(B_n) \ \vdash^? \ A$ does not succeed.
and so let $\Delta_n = \Delta_{n-1} \cup Cop(B_n)$.
Let $\Delta' = \bigcup_n \Delta_n$
Clearly $\Delta' \ \vdash^? \ A$ does not succeed.

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

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

by induction on $B$.

(a) For atoms this is the definition.
(b) Let $B = C \to D$. We prove the two directions by simultaneous induction.
(b1) Suppose $\Delta' \ \vdash^? \ C \to D$ succeeds. If $h(C) = \mathbf{false}$, then $h(C \to D) = \mathbf{true}$ and we are done. Thus, assume $h(C) = \mathbf{true}$. By the induction hypothesis, it follows that $\Delta' \ \vdash^? \ C$ succeeds. Since, by hypothesis we have that $\Delta', C \ \vdash^? \ D$ succeeds, by cut we obtain that $\Delta' \ \vdash^? \ D$ succeeds, and hence by the induction hypothesis $h(D) = \mathbf{true}$.
(b2) If $\Delta' \ \vdash^? \ C \to D$ does not succeed, we will show that $h(C \to D) = \mathbf{false}$. Let $Head(D) = q$, we get

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

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

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

From the above theorem and Lemma 2.26 we immediately obtain the completeness of the proof procedure with restart from the initial goal.

THEOREM 2.29. $\Delta \vdash A$ *in classical logic iff* $\Delta \vdash^? A, (A)$ *succeeds using the restart rule from the initial goal, added to the procedure of Definition 2.6.*

*Soundness and Completeness of the LL-procedure with Restart from any previous atomic Goal*

We now examine completeness for locally linear computation as in Definition 2.18 with the restart rule from any previous goal. As the next example shows, we cannot restrict the application of restart to the first atomic goal occurring in the computation.

EXAMPLE 2.30. The following query succeeds:

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

by the following computation:

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

It is clear that restarting from $r$, the first atomic goal would not help.

THEOREM 2.31. *[Soundness and completeness of locally linear computation with restart]*
$\Delta \vdash^? G, H$ *succeeds iff* $\Delta \vdash G \vee \bigvee H$ *in classical logic.*

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

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

2. *Length* $k+1$

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

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

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

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

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

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

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

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

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

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

It is well-known that the relation $\prec$ is a well-order on integer multisets [Dershowitz and Manna, 1979].

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

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

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

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

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

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

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

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

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

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

## *Some Efficiency Consideration*

The proof systems based on locally linear computation are a good starting point for designing efficient automated-deduction procedures; on the one hand proof search is guided by the goal, on the other hand derivations have a smaller size since a formula that has to be reused does not create further branching. We now want to remark upon termination of the procedures.

The basic LL- procedure obviously terminates: since formulas are thrown out as soon as they are used in a reduction step, every branch of a given derivation eventually ends with a query which either immediately succeeds, or no further reduction step is possible from it. This was the motivation of the LL- procedure as an alternative to a loop-checking mechanism. Does the (bounded) restart rule preserve this property? As we have stated, it does not, in the sense that a silly kind of loop may be created by restart. Let us consider the following example, here we give the computation for intuitionistic logic, but the example works for the classical case as well:

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

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

This is a loop created by restart. It is clear that continuing the derivation by restart does not help, as none of the new atomic goals match the head of any formula in the database.

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

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

step to

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

It is obvious that this restriction preserves completeness.

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

EXAMPLE 2.32. Let $\Delta$ contain the following formulas:

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

Then, we have the following deduction:

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

Then we get

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

The first query succeeds, the latter is reduced to

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

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

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

We now apply bounded restart on $s$.

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

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

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

$$\begin{aligned} Acc(H, q_i, q_j) \quad \equiv \quad & \exists q_k \in H \ (k < j \wedge q_k = q_i) \vee \\ & \vee \exists q_k \in H(Acc(H, q_i, q_k) \wedge Acc(H, q_k, q_j)). \end{aligned}$$

The modified bounded restart rule for intuitionistic logic becomes: from

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

step to

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

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

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

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

LEMMA 2.33. *Let*

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

*and*

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

*Then (a) is equivalent to (b):*

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

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

By the previous lemma we immediately have.

PROPOSITION 2.34. *In any computation of* $\Delta \vdash^? q, H$ *with restart, no backtracking is necessary. The atom* $q$ *can match with the head of any* $A_1 \to \ldots \to A_n \to q \in \Delta$ *and success or failure does not depend on the choice of such a formula.*

The parallel property to Lemma 2.33, Proposition 2.34 does not hold for the intuitionistic case. Let $A = A_1 \to \ldots \to A_n \to q$, $B = B_1 \to \ldots \to B_m \to q$. Then (a) is *not necessarily* equivalent to (b):

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

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

To this regard, let

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

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

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

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

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

The first computation fails (we cannot restart here) and the second computation succeeds. Trivially, $\{c, a, a \to q, b \to q\}$ does not prove $b$, whereas it does prove $a$. We thus see that in the intuitionistic computation $\Delta \vdash^? q, H$, with bounded restart, backtracking is certainly necessary. The atom $q$ can unify with any formula $A_1 \to \ldots \to A_n \to q$ and success or failure may depend on the choice of the formula. This may give an intuitive account of the difference of complexity between the intuitionistic and the classical case.[2]

## 4 EXTENDING THE LANGUAGE

### *Conjunction and negation*

In this and the next section we extend the language to the full propositional language. The addition of conjunction to the propositional language does not change

[2]We recall that intutionistic provability is PSPACE-complete [Statman, 1979], whereas classical provability is CoNP-complete.

the system much. As we have seen at the beginning of the chapter, $\wedge$ can be fully characterised by the following two conditions on the consequence relation:

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

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

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

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

LEMMA 2.35.

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

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

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

*4.* $(A \to B) \wedge (A \to C) \vdash A \to B \wedge C$. [3]

**Proof.** Exercise. ■

LEMMA 2.36. *Every formula $A$ with conjunctions is equivalent in intuitionistic logic to a conjunction of formulas which contain no conjunctions.*

**Proof.** Use the equivalences of Lemma 2.35 to pull the conjunctions out. ■

DEFINITION 2.37.

1. With any formula $A$ with conjunctions we associate a unique (up to equivalence) set $C(A)$ of formulas as follows: given $A$, use Lemma 2.36 to present it as $\wedge A_i$, where each $A_i$ contains no conjunctions. Let $C(A) = \{A_i\}$.

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

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

$$C(\Delta) \ \vdash^? \ B \quad \forall B \in C(A).$$

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

[3] As a notational convention, we assume that $\wedge$ is more binding than $\to$.

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

DEFINITION 2.38. [Computations for data and goal containing $\bot$ for intuitionistic and classical logic] The basic procedure is that one defined in 2.6, (plus the restart rule for classical logic), with the following modifications:

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

$$\Delta \vdash^? B_i$$

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

In Definition 2.38 we have actually defined two procedures. One is the computation without the restart rule for intuitionistic logic with $\bot$, and the other is the computation with the restart rule for classical logic. We have to show that the two procedures indeed correctly capture the intended fragment of the respective systems. This is easy to see. The rule $\bot \vdash A$ is built into the computation via the modifications in 1. and 2. of Definition 2.38 and hence we know we are getting intuitionistic logic. To show that the restart rule yields classical logic, it is sufficient to show that the computation

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

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

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

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

The proof is similar to that one of Lemma 2.26, namely $Cop(G)$ is a way of representing $G \rightarrow \bot$ without using $\bot$.[4]

PROPOSITION 2.39. *For any database $\Delta$ and goal $G$:*

[4]Recently, this result appears in [Nadathur, 1998] where a uniform proof system for (first-order) classical logic is given.

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

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

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

for any formula $A$. The soundness of this generalisation may be easily derived from Proposition 2.39. Details are left to the reader.

EXAMPLE 2.40. We check:

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

By the implication rule

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

We apply reduction wrt. $(a \to b) \to b$.

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

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

We apply reduction wrt. $a \to\bot$

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

success.

EXAMPLE 2.41. We check:

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

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

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

We cannot use the reduction rule here. So, we fail in intuitionistic logic. In classical logic we can use restart to obtain:

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

and terminate successfully.

The locally linear computation with bounded restart (respectively restart) is complete for the $(\to, \bot, \wedge)$-fragment of intuitionistic (classical) logic. Here we show an example for intuitionistic logic.

EXAMPLE 2.42.

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

We show a derivation below. Steps (i) and (ii) are obtained by the reduction rule in the special case of $\bot$. Step (iii) is obtained by bounded restart. The explanation of the other steps is left to the reader.

$$\begin{array}{rcl}
(((a \to \bot) \to \bot) \to \bot) \to c & \vdash^? & (a \to \bot) \to (b \to \bot) \to c,\ \emptyset \\
(((a \to \bot) \to \bot) \to \bot) \to c, a \to \bot & \vdash^? & (b \to \bot) \to c,\ \emptyset \\
(((a \to \bot) \to \bot) \to \bot) \to c, a \to \bot, b \to \bot & \vdash^? & c,\ \emptyset \\
a \to \bot, b \to \bot & \vdash^? & ((a \to \bot) \to \bot) \to \bot,\ (c) \\
a \to \bot, b \to \bot, (a \to \bot) \to \bot & \vdash^? & \bot,\ (c) \\
a \to \bot, (a \to \bot) \to \bot & \vdash^? & b,\ (c, \bot) \\
(\mathbf{i})\ (a \to \bot) \to \bot & \vdash^? & a,\ (c, \bot, b) \\
(\mathbf{ii}) & \vdash^? & a \to \bot,\ (c, \bot, b, a) \\
a & \vdash^? & \bot,\ (c, \bot, b, a) \\
(\mathbf{iii})\ a & \vdash^? & a,\ (c, \bot, b, a, \bot) \\
 & & success.
\end{array}$$

## *Disjunction*

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

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

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

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

where $\equiv$ denotes logical equivalence in classical logic. 1. is valid in intuitionistic logic but 2. is not valid. In fact, if we add 2. as an axiom schema to intuitionistic logic we obtain a stronger logical system known as **LC** (see Chapter 4).

In intuitionistic logic we have the disjunction property, namely $\vdash_{\mathbf{I}} A \vee B$ iff $\vdash_{\mathbf{I}} A$ or $\vdash_{\mathbf{I}} B$. This is not true in classical logic **C**. Thus, for example, $\vdash_{\mathbf{C}} A \vee (A \to B)$ but $\nvdash_{\mathbf{C}} A$ and $\nvdash_{\mathbf{C}} A \to B$. We have seen at the beginning of the chapter, in Definition 2.4 the consequence relation rules defining disjunction. In view of those rules we may want to adopt the computation rule below for disjunction in the goal.

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

This corresponds to the consequence relation rules (4a) and (4b) of Definition 2.4.
In case we have disjunction in the data, the rule is clear:

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

This corresponds to the consequence relation rule (4c) of Definition 2.4. Let us adopt the above two rules for computation.

EXAMPLE 2.43. $A \vee B \ \vdash^? \ A \vee B$. Using R1 we get

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

or

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

which fail. However using R2, we get

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

which succeed using R1.

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

DEFINITION 2.44. Computation rules for full intuitionistic logic with disjunction.

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

| | |
|---|---|
| (suc) | $\Delta \ \vdash^? \ q$ succeeds if $q \in \Delta$ or $\bot \in \Delta$; |
| (conj) | from $\Delta \ \vdash^? \ A \wedge B$ step to $\Delta \ \vdash^? \ A$ and to $\Delta \ \vdash^? \ B$; |
| (g-dis) | from $\Delta \ \vdash^? \ A \vee B$ step to $\Delta \ \vdash^? \ A$ or to $\Delta \ \vdash^? \ B$; |
| (imp) | from $\Delta \ \vdash^? \ A \rightarrow B$ step to $\Delta + A \ \vdash^? \ B$; |
| (red) | from $\Delta \ \vdash^? \ G$ if $G$ is an atom $q$ or $G = A \vee B$, if $C \in \Delta$, with $C = A_1 \rightarrow \ldots A_n \rightarrow D$ (where $D$ is not an implication) step to |

(a) $\Delta \vdash^? A_i$, for $i = 1, \ldots, n$, and to
(b) $\Delta + D \vdash^? G$.

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

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

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

Similarly, even if the goal is an atom $q$ in the reduction case (red) we cannot require that $D$ in the formula $A_1 \to \ldots A_n \to D$ is atomic and $D = q$. If there are disjunctions in the database, at every step we can choose to work on the goal or to split the database by (c-dis) rule. Moreover, if there are $n$ formulas a systematic application of (c-dis) rule yields $2^n$ branches. All of this means that if we handle disjunction in the most obvious way we loose the goal-directedness of deduction and the computation becomes very inefficient. Can we do better? In the following we give an intuitionistic goal-directed procedure for data containing disjunction.

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

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

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

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

DEFINITION 2.45. A D-formula is defined as follows

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

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

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

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

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

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

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

DEFINITION 2.46. A (lablled) query has the form

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

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

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

where $x_i$ are labels and $D_i$ are D-formulas.

The proof rules for constraints are the following:

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

The computation rules are the following:

- (success-atom) $\Gamma \ \vdash^{?} \ x : q, H$ succeeds if for some $y$, $\Gamma \vdash y \leq x$, and either $y : q \in \Gamma$, or $y : \bot \in \Gamma \cup H \cup \{x\}$;
- (implication)

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

  step to

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

from $\Gamma \ \vdash^? \ x : \bigvee_{j=1}^{n} E_j, H$ step to
$\Gamma \ \vdash^? \ x : E_l, H \cup \{(x, \bigvee_{j \neq l}^{n} E_l)\}$ for some $l = 1, \ldots, n$;

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

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

- (restart) from

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

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

  $\Delta \ \vdash^? \ y : D', H \cup \{(x, q)\}$;

- (reduction-false) From $\Gamma \ \vdash^? \ x : q, H$ step to $\Gamma \ \vdash^? \ y : \bot, H$ for any $y \in \Gamma$.

Notice that the success-rule is a degenerate case of the reduction rule, (we have kept it distinct by imposing the constraints on the number of antecedents and consequents). Notice also the presence of the reduction-false rule. This rule is necessary in cases such as the following (if we allow them):

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

here the success rule does not help. However, we do not need the reduction-false rule, if the starting database does not prove $x : \bot$ for any $x$.

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

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

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

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

In general we cannot fix this $z$ a priori. It must be a $\leq$-minimal $z$ such that (1) succeeds. In specific cases we can determine $z$ in advance: if $n = 1$ and $E_n$ is atomic, we can always take $z = x$. If $m = 0$, we can always take $z = y$. [5]

EXAMPLE 2.47. Let $\Delta = \{x : A_1, \ldots, x : A_5\}$ where:

[5] In the general case we can postpone the choice of $z$ *at the end* of the computation. The technique is rather standard. To give a (simplified) hint, we consider $z$ as a variable $Z$; when an atomic goal $Z : r$ is reached and the database contains, say, $u : r$ we temptatively instantiate $Z$ with $u$ and then we check the constraints $y \leq u$ and $u \leq x$. If they are satisfied we are done. Otherwise, we try another instantiation, is any. Thus, each derivation generates a system of constraints/inequalities to be solved.

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

In Figure 2.3 it is shown a derivation of

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

which correspond to check $A_1 \wedge \ldots \wedge A_5 \vdash a \to (t \vee s)$ in intuitionistic logic. We adopt the follwing abbreviations: we omit $\Delta$ and in each node we only show the additional data whenever is introduced in the database (thus, the complete database in each query is given by $\Delta$ plus the formulas in each database on the branch from the root to that query). Moreover we omit the history as it is clear from the structure of the tree, we make an exception for $(y : t)$ which is used in the (unique) restart step. Here is an explanation of the steps: (4) is obtained by reduction wrt. $A_3$, (6) by reduction wrt. $A_2$. (7) and (8) are obtained by reduction wrt. $A_1$. (7) succeeds as $y : a$ has been put in the data on step (2). (9) and (10) are obtained by reduction wrt. $(b \to c) \vee (d \to e)$. (11) is obtained by reduction wrt. $b \to c$ and it succeeds as $z : b$ has been put in the data on step (5). (12) is obtained by restart. (13) is obtained by reduction wrt. $A_5$. (15) is obtained by reduction wrt. $d \to e$. (16) is obtained by reduction wrt. $A_4$ and it succeeds as $u : q$ has been put in the data on step (14).

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

DEFINITION 2.48. Given $\Delta$ as above, let $M = (W, \leq_M, w_0, V)$ be a Kripke model. A *realization* of $\Gamma$ is a mapping $\rho : \mathcal{A} \to W$, such that

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

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

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

THEOREM 2.49 (Soundness). *If* $\Delta \vdash x : E, H$ *is derivable then is valid.*

**Proof.** By induction on the length of computations. Details are left to the reader as an exercise. ■

In order to prove the completeness we need to show that the cut-rule, suitably formulated, is admissible.

By $\Gamma[x : D]$, we denote that $x : D \in \Gamma$.

THEOREM 2.50 (Cut for disjunctive databases). *If* $\Gamma[x : D] \vdash^? y : D_1, H_1$ *and* $\Delta \vdash^? z : D, H_2$ *succeeds,then also*

$(1)\ \vdash^? x : a \to t \vee s$

$(2)\ x \leq y, y : a \vdash^? y : t \vee s$

$(3)\ \vdash^? y : t,\ (\mathbf{y} : \mathbf{s})$

$(4)\ \vdash^? y : b \to p$

$(5)\ y \leq z, z : b \vdash^? z : p$

$(6)\ \vdash^? z : c$

$(7)\ \vdash^? y : a$

$(8)\ y : (b \to c) \vee (d \to e) \vdash^? z : c$

$(9)\ y : b \to c \vdash^? z : c$

$(10)\ y : d \to e \vdash^? z : c$

$(11)\ y : b \to c \vdash^? z : b$

$(12)\ \vdash^? y : s$ **(restart)**

$(13)\ \vdash^? y : q \to e$

$(14)\ y \leq u, u : q \vdash^? u : e$

$(15)\ \vdash^? u : d$

$(16)\ \vdash^? u : q$

Figure 2.3. Derivation for Example 2.47.

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

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

THEOREM 2.51. *If* $\Gamma \vdash^? x : D, H$ *is valid then it succeeds.*

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

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

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

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

- (step 0) Let $\Gamma_0 = \Gamma$, $H_0 = H$, $(y_0, E_0) = (x, D)$.
- (step n+1) Given $(y_n, E_n)$, we let
  - $\Gamma_{n+1} = \Gamma_n$,
  - $(y_{n+1}, E_{n+1}) = (x_{k+1}, D_{k+1})$, where $k = \max_{t \leq n} \exists s_{\leq n} (y_s, E_s) = (x_t, D_t)$,
  - $H_{n+1} = H_n$,

  unless $y_n \in Lab(\Gamma_n)$ and we are in one of the cases listed below.

  - $E_n = q$ ($q$ is an atom) or $E_n = \bigvee_j^m G_j$, and $\Gamma_n \vdash^? y_n : E_n, H_n$ fails, in this case we let $H_{n+1} = H_n \cup \{(y_n, E_n)\}$.
  - $E_n = \bigwedge_{j=1}^m F_j \to D'$ and $\Gamma_n \vdash^? y_n : E_n, H_n$ fails, we let
    $$\Gamma_{n+1} = \Gamma_n \cup \{y_n \leq x_s, x_s : F_1, \ldots, x_s : F_m\},$$
    $$(y_{n+1}, E_{n+1}) = (x_s, D'),$$
    where $x_s = \min\{x_t \in \mathcal{A} \mid x_t \notin Lab(\Gamma_n)\}$.
  - $E_n = \bigvee_j^m G_j$, and $\Gamma_n \vdash^? y_n : E_n, H_n$ succeeds, but

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

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

Each stage of the construction defines, so to say, a new query. Intuitively, we follow the enumeration $(x_n, D_n)$ in order to determine which formula and world to consider at the next stage, unless the formula currently evaluated is implicational (and it fails). In such a case, we evaluate its consequent at a newly created world. When we come to an atomic formula, or to a disjunction, we go back to the enumeration $(x_n, D_n)$ to pick the next pair.

We then define an intuitionistic Kripke-model as follows $M = (W, u_0, \leq, V)$, such that

- $W = \bigcup_n Lab(\Gamma_n) \cup \{u_0\}$, where $u_0 \notin \bigcup_n \Gamma_n$;
- $x \leq y \equiv x = u_0 \vee \exists n \ \Gamma_n \vdash x \leq y$;
- $V(u_0) = \emptyset$;
- $V(x) = \{q \mid \exists n \ x \in Lab(\Gamma_n) \ \wedge \ \Gamma_n \ \vdash^? \ x : q, ; H_n$ succeeds$\}$ for $x \neq u_0$.

It is easy to see that $\leq$ is reflexive and transitive and that $V$ is monotonic with respect to $\leq$. We show that for all $x \in W$, $x \neq u_0$ and D-formulas $E$,

$$M, x \models E \ \Leftrightarrow \ \exists n \ x \in Lab(\Gamma_n) \wedge \Gamma_n \ \vdash^? \ x : E, H_n \text{ succeeds.}$$

We are able now to conclude the proof of the completeness theorem. Let $\rho(z) = z$, for every $z \in Lab(\Gamma)$, where $\Gamma$ is the original database. It is easy to see that $\rho$ is a realization of $\Gamma$ in $M$ and if $u : C \in \Gamma$, then by identity and the previous fact we have $M, \rho(u) \models C$. On the other hand, we have $M, \rho(x) \not\models D$ and for all $(y, E) \in H$, $M, \rho(y) \not\models E$. This concludes the proof. ■

# CHAPTER 3

# MODAL LOGICS OF STRICT IMPLICATION

## 1 INTRODUCTION

We study how to extend the goal-directed proof methods to strict implication modal logics. We consider this as a first step in order to extend the goal-directed paradigm to the realm of modal logics. Strict implication, denoted by $A \Rightarrow B$ is read as 'necessarily $A$ implies $B$'. The notion of necessity (and the dual notion of possibility) are the subject of modal logics. Strict implication can be regarded as a derived notion: $A \Rightarrow B = \Box(A \rightarrow B)$, where $\rightarrow$ denotes material implication and $\Box$ denotes modal necessity. However, strict implication can also be considered as a primitive notion, and has already been considered as such at the beginning of the century in many discussions about the paradoxes of material implication [Lewis, 1912; Lewis and Langford, 1932].

The extension of the goal-directed approach to strict implication and modal logics relies upon the *possible worlds* semantics of modal logics which is mainly due to Kripke. As we have already done in the case of intuitionistic and intermediate logics, we regard a database as a set of labelled formulas $x_i : A_i$ equipped by a relation $\alpha$ giving connections between labels. The labels represent worlds, states, positions. Thus, $x_i : A_i$ means that $A_i$ holds at $x_i$. The goal of a query is always asked with respect to a position/world. The form of databases and goals determine the notion of consequence relation

$$\{x_1 : A_1, \ldots, x_n : A_n\}, \alpha \vdash x : A$$

whose intended meaning is that if $A_i$ holds at $x_i$ (for $i = 1, \ldots, n$) and the $x_i$ are connected as $\alpha$ prescribes, then $A$ must hold at $x$.

For different logics $\alpha$ will be required to satisfy different properties such as reflexivity, transitivity, etc., depending on the properties of the accessibility relation of the system under consideration.

In most of the chapter we will be concerned with implicational modal logics whose language $\mathcal{L}(\Rightarrow)$ contains all formulas built out from a denumerable set *Var* of propositional variables by applying the strict implication connective, that is, if $p \in$ *Var* then $p$ is a formula of $\mathcal{L}(\Rightarrow)$, and if $A$ and $B$ are formulas of $\mathcal{L}(\Rightarrow)$, then so is $A \Rightarrow B$. Let us fix an atom $p_0$, we can define the constant $\top \equiv p_0 \Rightarrow p_0$ and let $\Box A \equiv \top \Rightarrow A$.

**Semantics**
We review the standard Kripke semantics for $\mathcal{L}(\Rightarrow)$.

Table 3.1. Standard properties of the accessilbility relation.

| | |
|---|---|
| Reflexivity | $\forall x\ xRx$ |
| Transitivity | $\forall x \forall y \forall z\ xRy \wedge yRz \to xRz$ |
| Symmetry | $\forall x \forall y\ xRy \to yRx$ |
| Euclidean | $\forall x \forall y \forall z\ xRy \wedge xRz \to yRz$ |

A Kripke structure $M$ for $\mathcal{L}(\Rightarrow)$ is a triple $(W, R, V)$, where $W$ is a non-empty set (whose elements are called possible worlds), $R$ is a binary relation on $W$, and $V$ is a mapping from $W$ to sets of propositional variables of $\mathcal{L}$. Truth conditions for formulas (of $\mathcal{L}(\Rightarrow)$) are defined as follows:

- $M, w \models p$ iff $p \in V(w)$;
- $M, w \models A \Rightarrow B$ iff for all $w'$ such that $wRw'$ and $M, w' \models A$, it holds $M, w' \models B$.

We say that a formula $A$ is *valid* in a structure $M$, denoted by $M \models A$, if $\forall w \in W,\ M, w \models A$. We say that a formula $A$ is *valid* with respect to a given class of structures $\mathcal{K}$, iff it is valid in every structure $M \in \mathcal{K}$. We sometimes use the notation $\models_{\mathcal{K}} A$. Let us fix a class of structures $\mathcal{K}$. Given two formulas $A$ and $B$, we can define the *consequence relation* $A \models_{\mathcal{K}} B$ as

$$\forall M \in \mathcal{K}\ \forall w \in W \quad \text{if} \quad M, w \models_{\mathcal{K}} A \text{ then } M, w \models_{\mathcal{K}} B.$$

Different modal logics are obtained by considering classes of structures whose relation $R$ satisfies some specific properties. The properties of the accessibility relations we consider are listed in Figure 3.1.

We will take into consideration strict implication $\Rightarrow$ as defined in systems **K**, **KT**,[1] **K4**, **S4**, **K5**, **K45**, **KB**, **KBT**, and **S5**. [2]

Properties of accessibility relation $R$ in Kripke frames, corresponding to these systems are shown in Figure 3.2

Letting **S** be one of the modal systems above, we use the notation $\models_{\mathbf{S}} A$ (and $A \models_{\mathbf{S}} B$) to denote validity in (and the consequence relation determined by) the class of structures corresponding to **S**.

## 2 PROOF SYSTEMS

In this section we present proof methods for all modal systems mentioned above.

[1]We use the acronym **KT** rather than the more common **T**, as the latter is also the name of a subrelevance logic we will meet in Chapter 5.

[2]We do not consider here systems containing $D : 2A \to 3A$, which correspond to the *seriality* of the accessibility relation, i.e. $\forall x \exists y\ xRy$ in Kripke frames. The reason is that seriality cannot be expressed in the language of strict implication alone; moreover, it cannot be expressed in any modal language, unless $\neg$ or $3$ is allowed. We will come back to the treatment of seriality in Section 4.

Table 3.2. Some standard modal logics.

| Name | Reflexivity | Transitivity | Symmetry | Euclidean |
|---|---|---|---|---|
| **K** | | | | |
| **KT** | * | | | |
| **K4** | | * | | |
| **S4** | * | * | | |
| **K5** | | | | * |
| **K45** | | * | | * |
| **KB** | | | * | |
| **KTB** | * | | * | |
| **S5** | * | * | * | * |

DEFINITION 3.1. Let us fix a denumerable alphabet $\mathcal{A} = \{x_1, \ldots, x_i, \ldots\}$ of labels. A *database* is a finite graph of formulas labelled by $\mathcal{A}$. We denote a database as a pair $(\Delta, \alpha)$, where $\Delta$ is a finite set of labelled formulas $\Delta = \{x_1 : A_1, \ldots, x_n : A_n\}$ and $\alpha = \{(x_1, x'_1), \ldots, (x_m, x'_m)\}$ is a set of links. Let *Lab*$(E)$ denote the set of labels $x \in \mathcal{A}$ occurring in $E$, and assume that (i) *Lab*$(\Delta) =$ *Lab*$(\alpha)$, and (ii) if $x : A \in \Delta$, $x : B \in \Delta$, then $A = B$.[3]

A *trivial database* has the form $(\{x_0 : A\}, \emptyset)$.

The *expansion* of a database $(\Gamma, \alpha)$ by $y : C$ at $x$, with $x \in$ *Lab*$(\Gamma)$, $y \notin$ *Lab*$(\Gamma)$ is defined as follows:

$$(\Gamma, \alpha) \oplus_x (y : C) = (\Delta \cup \{y : C\}, \alpha \cup \{(x, y)\}).$$

DEFINITION 3.2. A query $Q$ is an expression of the form:

$$Q = \ (\Delta, \alpha) \ \vdash^? \ x : G, H$$

where $(\Delta, \alpha)$ is a database, $x \in$ *Lab*$(\Delta)$, $G$ is a formula, and $H$, *the history*, is a set of pairs

$$H = \{(x_1, q_1), \ldots, (x_m, q_m)\}$$

where $x_i$ are labels and $q_i$ are atoms. We will often omit the parentheses around the two components of a database and write $Q = \Delta, \alpha \ \vdash^? \ x : G, H$. A query from a trivial database $\{x_0 : A\}$ will be written simply as:

$$x_0 : A \ \vdash^? \ x_0 : B, H,$$

and if $A = \top$, we sometimes just write $\vdash^? \ x_0 : B, H$.

DEFINITION 3.3. Let $\alpha$ be a set of links, we introduce a family of relation symbols $A^{\mathbf{S}}_{\alpha}(x, y)$, where $x, y \in$ *Lab*$(\alpha)$. We consider the following conditions:

[3] We will drop this condition in Section 4 when we extend the language by allowing conjunction.

**(K)** $(x, y) \in \alpha \;\Rightarrow\; A^{\mathbf{S}}_{\alpha}(x, y)$,
**(T)** $x = y \;\Rightarrow\; A^{\mathbf{S}}_{\alpha}(x, y)$,
**(4)** $\exists z(A^{\mathbf{S}}_{\alpha}(x, z) \wedge A^{\mathbf{S}}_{\alpha}(z, y)) \;\Rightarrow\; A^{\mathbf{S}}_{\alpha}(x, y)$,
**(5)** $\exists z(A^{\mathbf{S}}_{\alpha}(z, x) \wedge A^{\mathbf{S}}_{\alpha}(z, y)) \;\Rightarrow\; A^{\mathbf{S}}_{\alpha}(x, y)$,
**(B)** $A^{\mathbf{S}}_{\alpha}(x, y) \;\Rightarrow\; A^{\mathbf{S}}_{\alpha}(y, x)$.

For $\mathbf{K} \in \mathbf{S} \subseteq \{\mathbf{K}, \mathbf{T}, \mathbf{4}, \mathbf{5}, \mathbf{B}\}$, we let $A^{\mathbf{S}}$ be the least relation satisfying all conditions in **S**. Thus, for instance, $A^{\mathbf{K45}}$ is the least relation such that:

$$\begin{aligned} A^{\mathbf{K45}}_{\alpha}(x, y) \quad \Leftrightarrow \quad & (x, y) \in \alpha \vee \\ & \vee \exists z(A^{\mathbf{K45}}_{\alpha}(x, z) \wedge A^{\mathbf{K45}}_{\alpha}(z, y)) \vee \\ & \vee \exists z(A^{\mathbf{K45}}_{\alpha}(z, x) \wedge A^{\mathbf{K45}}_{\alpha}(z, y)). \end{aligned}$$

We will use the standard abbreviations (i.e. $A^{\mathbf{S5}} = A^{\mathbf{KT5}} = A^{\mathbf{KT45}}$).

DEFINITION 3.4 (Modal Deduction Rules). For each modal system **S**, the corresponding proof system, denoted by P(**S**), comprises the following rules parametrized to predicates $A^{\mathbf{S}}$:

- (success) $\Delta, \alpha \;\vdash^{?}\; x : q, H$ immediately succeeds if $q$ is an atom and $x : q \in \Delta$.
- (implication) From $\Delta, \alpha \vdash^{?} x : A \Rightarrow B, H$, step to

  $(\Delta, \alpha) \oplus_x (y : A) \vdash^{?} y : B, H,$

  where $y \notin \mathit{Lab}(\Delta) \cup \mathit{Lab}(H)$.
- (reduction) If $y : C \in \Delta$, with $C = B_1 \Rightarrow B_2 \Rightarrow \ldots \Rightarrow B_k \Rightarrow q$, with $q$ atomic, then from

  $\Delta, \alpha \vdash^{?} x : q, H$

  step to

  $\Delta, \alpha \vdash^{?} u_1 : B_1, H \cup \{(x, q)\}, \quad \ldots, \quad \Delta, \alpha \vdash^{?} u_k : B_k, H \cup \{(x, q)\},$

  for some $u_0, \ldots, u_k \in \mathit{Lab}(\alpha)$, with $u_0 = y$, $u_k = x$, such that

  for $i = 0, \ldots, k-1$, $A^{\mathbf{S}}_{\alpha}(u_i, u_{i+1})$ holds.
- (restart) If $(y, r) \in H$, then, from $\Delta, \alpha \vdash^{?} x : q, H$, with $q$ atomic, step to

  $\Delta, \alpha \vdash^{?} y : r, H \cup \{(x, q)\}.$

Since most of the results which follow do not depend on the specific properties of the predicates $A^{\mathbf{S}}$ involved in the definition of a proof system P(**S**), we will often omit the reference to P(**S**). The following proposition states easy properties of the deduction procedure.

PROPOSITION 3.5.

- *(Identity) If $x : A \in \Gamma$, then $\Gamma, \alpha \vdash^? x : A, H$ succeeds.*
- *(Monotony) If $Q = \Gamma, \alpha \vdash^? x : C, H$ succeeds and $\Gamma \subseteq \Delta$, $\alpha \subseteq \beta$, $H \subseteq H'$, then also*

  $Q' = \Delta, \beta \vdash^? x : C, H'$ *succeeds.*

  *Moreover, any derivation of $Q$ can be turned uniformly into a derivation of $Q'$ by replacing $\Gamma$ with $\Delta$, $\alpha$ with $\beta$, and $H$ with $H'$.*
- *(Increasingness) Let $\mathcal{D}$ be any derivation of a given query; if $Q_1 = \Gamma_1, \alpha_1 \vdash^? x_1 : A_1, H_1$ and $Q_2 = \Gamma_2, \alpha_2 \vdash^? x_2 : A_2, H_2$ are two queries in $\mathcal{D}$, such that $Q_2$ is a descendant of $Q_1$, then $\Gamma_1 \subseteq \Gamma_2$, $\alpha_1 \subseteq \alpha_2$, $H_1 \subseteq H_2$.*

**Restricted restart**
Similar to the case of classical logic, in any deduction of a query $Q$ of the form $\Delta, \alpha \vdash^? x : G, \emptyset$, the *restart rule* can be restricted to the choice of the pair $(y, r)$, such that $r$ is the uppermost atomic goal occurred in the deduction and $y$ is the label associated to $r$ (that is, the query in which $r$ appears contains $\ldots \vdash^? y : r$). Hence, if the initial query is $Q = \Delta, \alpha \vdash^? x : G, \emptyset$ and $G$ is an atom $q$, such a pair is $(x, q)$, if $G$ has the form $B_1 \Rightarrow \ldots \Rightarrow B_k \Rightarrow r$, then the first pair is obtained by repeatedly applying the implication rule until we reach the query $\ldots \vdash^? x_k : r$, with $x_k \notin \mathit{Lab}(\Delta)$. With this restriction, we do not need to keep track of the history any more, but only of the first pair. An equivalent formulation is to allow restart from the initial goal (and its relative label) even if it is implicational, but the re-evaluation of an implication causes a redundant increase of the database, that is why we have preferred the above formulation.

PROPOSITION 3.6. *If $\Delta, \alpha \vdash^? x : G, \emptyset$ succeeds then it succeeds by a derivation in which every application of restart is* restricted restart.

**Proof.** It suffices to show the following fact: whenever, in a successful derivation, we restart from a pair $(x, p)$ coming from a query $Q$ preceding the current one, we still obtain a successful derivation if we restart from any pair $(y, q)$ coming from a query $Q'$ preceding $Q$. This fact implies that if the initial query succeeds, then there is a successful derivation in which every restart application makes use only of the first pair. The proof of this fact is essentially identical to the one of the ($\Rightarrow$) half of Proposition 2.24. We omit the details. ■

$$(1)\ \vdash^{?}\ x_0 : ((p \Rightarrow p) \Rightarrow a \Rightarrow b) \Rightarrow (b \Rightarrow c) \Rightarrow a \Rightarrow c$$

$$(2)\ x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b,\ \alpha\ \vdash^{?}\ x_1 : (b \Rightarrow c) \Rightarrow a \Rightarrow c$$

$$(3)\ x_2 : b \Rightarrow c,\ \alpha\ \vdash^{?}\ x_2 : a \Rightarrow c$$

$$(4)\ x_3 : a,\ \alpha\ \vdash^{?}\ x_3 : c$$

$$(5)\ \alpha\ \vdash^{?}\ x_3 : b,\ (\mathbf{x_3}, \mathbf{c})$$

$$(6)\ \alpha\ \vdash^{?}\ x_2 : p \Rightarrow p,\ (\mathbf{x_3}, \mathbf{c}) \qquad (8)\ \alpha\ \vdash^{?}\ x_3 : a,\ (\mathbf{x_3}, \mathbf{c})$$

$$(7)\ x_4 : p, \alpha \cup \{(x_2, x_4)\}\ \vdash^{?}\ x_4 : p,\ (\mathbf{x_3}, \mathbf{c})$$

Figure 3.1. Derivation for Example 3.7.

EXAMPLE 3.7. In Figure 3.1 we show a derivation of

$$((p \Rightarrow p) \Rightarrow a \Rightarrow b) \Rightarrow (b \Rightarrow c) \Rightarrow a \Rightarrow c.$$

in P(**K**). By Proposition 3.6, we only record the first pair for restart, which, however, is not used in the derivation. As usual in each node we only show the additional data, if any. Thus the database in each node is given by the collection of the formulas from the root to that node. Here is an explanation of the steps: in step (2) $\alpha = \{(x_0, x_1)\}$; in step (3) $\alpha = \{(x_0, x_1), (x_1, x_2)\}$; in step (4) $\alpha = \{(x_0, x_1), (x_1, x_2), (x_2, x_3)\}$; since $A^{\mathbf{K}}_{\alpha}(x_2, x_3)$, by reduction w.r.t. $x_2 : b \Rightarrow c$ we get (5); since $A^{\mathbf{K}}_{\alpha}(x_1, x_2)$ and $A^{\mathbf{K}}_{\alpha}(x_2, x_3)$, by reduction w.r.t. $x_1 : (p \Rightarrow p) \Rightarrow a \Rightarrow b$ we get (6) and (8). the latter immediately succeeds as $x_3 : a \in \Delta$; from (6) we step to (7) which immediately succeeds.

EXAMPLE 3.8. In Figure 3.2 we show a derivation of

$$((((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p) \Rightarrow p$$

in P(**KBT**), we use restricted restart according to Proposition 3.6. In step (2), $\alpha = \{(x_0, x_1)\}$. Step (3) is obtained by reduction w.r.t. $x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p$, as $A^{\mathbf{KBT}}_{\alpha}(x_1, x_1)$. In step (4) $\alpha = \{(x_0, x_1), (x_1, x_2)\}$; step (5) is obtained by restart; step (6) by reduction w.r.t. $x_2 : (a \Rightarrow a) \Rightarrow p$, as $A^{\mathbf{KBT}}_{\alpha}(x_2, x_1)$; in step (7) $\alpha = \{(x_0, x_1), (x_1, x_2), (x_1, x_3)\}$ and the query immediately succeeds.

We are going to show that the cut rule is admissible for the deduction procedure. The cut rule states the following: let $x : A \in \Gamma$, then if (1) $\Gamma \vdash y : B$ and (2) $\Delta \vdash z : A$ succeed, we can 'replace' $x : A$ by $\Delta$ in $\Gamma$ and get a successful query from (1). There are two points to clarify. First, we need to define the involved notion of

$$
\begin{array}{c}
(1)\ \vdash^? \ x_0 : ((((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p) \Rightarrow p \\
| \\
(2)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, \alpha \ \vdash^? \ x_1 : p \\
| \\
(3)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p,\ \alpha \ \vdash^? \ x_1 : ((a \Rightarrow a) \Rightarrow p) \Rightarrow q,\ (\mathbf{x_1}, \mathbf{p}) \\
| \\
(4)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, x_2 : (a \Rightarrow a) \Rightarrow p,\ \alpha \ \vdash^? \ x_2 : q,\ (\mathbf{x_1}, \mathbf{p}) \\
| \\
(5)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, x_2 : (a \Rightarrow a) \Rightarrow p,\ \alpha \ \vdash^? \ x_1 : p,\ (\mathbf{x_1}, \mathbf{p}) \\
| \\
(6)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, x_2 : (a \Rightarrow a) \Rightarrow p,\ \alpha \ \vdash^? \ x_1 : a \Rightarrow a,\ (\mathbf{x_1}, \mathbf{p}) \\
| \\
(7)\ x_1 : (((a \Rightarrow a) \Rightarrow p) \Rightarrow q) \Rightarrow p, x_2 : (a \Rightarrow a) \Rightarrow p, x_3 : a, \alpha \ \vdash^? \ x_3 : a,\ (\mathbf{x_1}, \mathbf{p})
\end{array}
$$

Figure 3.2. Derivation for Example 3.8.

substitution, and this we do in the next definition. Furthermore the proof systems P(**S**) depend uniformly on predicate $A^{\mathbf{S}}$, and we expect that the admissibility of cut depends on the properties of predicate $A^{\mathbf{S}}$. It will turn out that the admissibility of cut (proved in Theorem 3.10) will hold for every proof system P(**S**), such that $A^{\mathbf{S}}$ satisfies the following conditions:

- (i) $A^{\mathbf{S}}$ is closed under substitution of labels;
- (ii) $A^{\mathbf{S}}_{\alpha}(x, y)$ implies $A^{\mathbf{S}}_{\alpha \cup \beta}(x, y)$;
- (iii) $A^{\mathbf{S}}_{\alpha}(u, v)$ implies $\forall x\ y\ (A^{\mathbf{S}}_{\alpha \cup \{(u,v)\}}(x, y) \leftrightarrow A^{\mathbf{S}}_{\alpha}(x, y))$.

We say that two databases $(\Gamma, \alpha)$, $(\Delta, \beta)$ are *compatible for substitution*,[4] if

> for every $x \in$ *Lab*$(\Gamma) \cap$ *Lab*$(\Delta)$, for all formulas $C$, $x : C \in \Gamma \Leftrightarrow x : C \in \Delta$.

If $(\Gamma, \alpha)$ and $(\Delta, \beta)$ are compatible for substitution, $x : A \in \Gamma$, and $y \in$ *Lab*$(\Delta)$, we denote by

$$(\Gamma, \alpha)[x : A/\Delta, \beta, y] = (\Gamma - \{x : A\} \cup \Delta, \alpha[x/y] \cup \beta).$$

the database which results by replacing $x : A$ in $(\Gamma, \alpha)$ by $(\Delta, \beta)$ at point $y$.

We say that a predicate $A^{\mathbf{S}}$ is *closed under substitution* if,

[4]We will drop this condition in Section 4, when we add conjunction.

whenever $A^{\mathsf{S}}_{\alpha}(x, y)$ holds, then $A^{\mathsf{S}}_{\alpha[u/v]}(x[u/v], y[u/v])$ also holds.

PROPOSITION 3.9.

*(a) If $A^{\mathsf{S}}$ satisfies condition (i) above and $\Gamma, \alpha \ \vdash^{?} \ x : C, H$ succeeds, then also*

$$\Gamma[u/v], \alpha[u/v] \ \vdash^{?} \ x[u/v] : C, H[u/v] \textit{ succeeds.}$$

*(b) If $A^{\mathsf{S}}$ satisfies condition (iii) above, then $A^{\mathsf{S}}_{\alpha}(x, y)$ and $\Gamma, \alpha \cup \{(x, y)\} \ \vdash^{?} \ u : G, H$ succeed, implies $\Gamma, \alpha \ \vdash^{?} \ u : G, H$ succeeds.*

THEOREM 3.10 (Admissibility of cut). *Let predicate $A^{\mathsf{S}}$ satisfy the conditions (i), (ii), (iii) above. If the following queries succeed in the proof system P(**S**) :*

*1.* $\Gamma[x : A] \ \vdash^{?} \ u : B, H_1$

2. $\Delta, \beta \ \vdash^{?} \ y : A, H_2$.

*and $(\Gamma, \alpha)$ and $(\Delta, \beta)$ are compatible for substitution, then also*

*3.* $(\Gamma, \alpha)[x : A/\Delta, \beta, y] \ \vdash^{?} \ u[x/y] : B, H_1[x/y] \cup H_2$ *succeeds in P(**S**) .*

COROLLARY 3.11. *Under the same conditions as above, if $x : A \ \vdash^{?} \ x : B$ succeeds and $x : B \ \vdash^{?} \ x : C$ succeeds then also $x : A \ \vdash^{?} \ x : C$ succeeds.*

COROLLARY 3.12. *If $\mathbf{K} \in \mathbf{S} \subseteq \{\mathbf{K}, \mathbf{4}, \mathbf{5}, \mathbf{B}, \mathbf{T}\}$, then in the proof system P(**S**) cut is admissible.*

**Proof.** One can easily check that the predicate $A^{\mathsf{S}}$ satisfies the conditions of the previous theorem. ■

## *2.1 Soundness and Completeness*

We need to give a semantic meaning to queries. We do this by introducing a suitable notion of realization, similarly to what we did in the cases of intuitionistic and intermediate logic (see Definition 2.48).

DEFINITION 3.13 (Realization). Let $A^{\mathsf{S}}$ be an accessibility predicate, given a database $(\Gamma, \alpha)$ and a Kripke model $M = (W, R, V)$, a mapping $f$ : *Lab*$(\Gamma) \rightarrow W$ is called a *realization* of $(\Gamma, \alpha)$ in $M$ with respect to $A^{\mathsf{S}}$, if the following hold:

1. $A^{\mathsf{S}}_{\alpha}(x, y)$ implies $f(x)Rf(y)$;
2. if $x : A \ \in \Gamma$, then $M, f(x) \models A$.

We say that a query $Q = \Gamma \ , \alpha \ \vdash^{?} \ x : G, H$ is *valid* in **S** if for every **S**-model $M$ and every realization $f$ of $(\Gamma, \alpha)$, we have either $M, f(x) \models G$, or for some $(y, r) \in H$, $M, f(y) \models r$.

THEOREM 3.14 (Soundness). *Let $Q = \Gamma, \alpha \ \vdash^{?} \ x : G, H$ succeed in the proof system P(**S**) , then it is valid in **S**.*

**Proof.** Let $M = (W, R, V)$ be an **S**-model and $f$ be a realization of $(\Gamma, \alpha)$ in $M$, we proceed by induction on the height $h$ of a successful derivation of $Q$. For the base case, we have that $Q$ immediately succeeds, $G$ is atomic, and $x : q \in \Delta$; by hypothesis $M, f(x) \models q$.

In the induction step, we have several cases. Let the first step in the derivation be obtained by *implication*, then $G = A \Rightarrow B$. Suppose by way of contradiction that $M, f(x) \not\models A \Rightarrow B$ and for all $(y, r) \in H$, $M, f(y) \not\models r$. We have that for some $w \in W$, such that $f(x)Rw$, $M, w \models A$, but $M, w \not\models B$. From $Q$ we step to

$$Q' = (\Gamma, \alpha) \oplus_x (u : A) \vdash^? u : B, H, \text{with } u \notin \mathit{Lab}(\Delta) \cup \mathit{Lab}(H),$$

which succeeds by a smaller derivation. Let $f'(z) = f(z)$, for $z \neq u$ and $f'(u) = w$. Then, $f'$ is a realization of $(\Gamma, \alpha) \oplus_x (u : A)$, and by the induction hypothesis, either $M, f'(u) \models B$, whence $M, w \models B$, or for some $(y, r) \in H$, $M, f'(y) \models r$, whence $M, f(y) \models r$; in both cases we have contradictions.

Let the first step in the derivation be obtained by *reduction*, then $G$ is an atom $q$, there is $z : C \in \Delta$, with $C = B_1 \Rightarrow \ldots \Rightarrow B_k \Rightarrow q$, and from $Q$ we step to

$$\Delta, \alpha \vdash^? u_1 : B_1, H \cup \{(x, q)\}, \quad \ldots, \quad \Delta, \alpha \vdash^? u_k : B_k, H \cup \{(x, q)\},$$

for some $u_0, \ldots, u_k \in \mathit{Lab}(\alpha)$, with $u_0 = z$, $u_k = x$, such that

for $i = 0, \ldots, k-1$, $A^{\mathbf{S}}_{\alpha}(u_i, u_{i+1})$ holds.

By hypothesis, we have

1. $M, f(z) \models B_1 \Rightarrow \ldots \Rightarrow B_k \Rightarrow q$, and
2. $f(u_i) R f(u_{i+1})$, for $i = 0, \ldots, k$.

By the induction hypothesis, either (a) for some $(y, r) \in H$, $M, f(y) \models r$, or (b) for $i = 1, \ldots, k$, $M, f(u_i) \models B_i$. In case (a) we are done. Suppose (b) holds. From $u_0 = z$, (1) and (2), we get

$$M, f(u_i) \models B_{i+1} \Rightarrow \ldots \Rightarrow B_k \Rightarrow q, \text{ for } i = 1, \ldots, k-1,$$

and finally $M, f(u_k) \models q$, that is $M, f(x) \models q$.

If the first step in the derivation is obtained by restart, then the claim immediately follows by the induction hypothesis. ■

COROLLARY 3.15. *If $x_0 : A \vdash^? x_0 : B, \emptyset$ succeeds in P(**S**), then $A \models_{\mathbf{S}} B$ holds. In particular, if $\vdash^? x_0 : A, \emptyset$ succeeds in P(**S**), then $A$ is valid in* **S**.

THEOREM 3.16 (Completeness). *Given a query $Q = \Gamma, \alpha \vdash^? x : A, H$, if $Q$ is* **S**-*valid then $Q$ succeeds in the proof system P(**S**).*

**Proof.** By contraposition, we prove that if $Q = \Gamma, \alpha \vdash^? x : A, H$ does not succeed in one proof system P(**S**), then there is an **S**-model $M$ and a realization $f$ of $(\Gamma, \alpha)$, such that $M, f(x) \not\models A$ and for any $(y, r) \in H$, $M, f(y) \not\models r$. The proof is very similar to the one of Theorem 2.51 for the completeness of the intuitionistic procedure with disjunction.

As usual, we construct an **S**-model by extending the database, through the evaluation of all possible formulas at every world (each represented by one label) of the database. Since such evaluation may lead, for implication formulas, to create new worlds, we must carry on the evaluation process on these new worlds. Therefore, in the construction we consider an enumeration of pairs $(x_i, A_i)$, where $x_i$ is a label and $A_i$ is a formula.

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

$$(x_0, A_0) = (x, A),$$
$$\forall y \in \mathcal{A}, \forall F \in \mathcal{L}, \forall n\, \exists m > n\; (y, F) = (x_m, A_m).$$

Given such enumeration we define i) a sequence of databases $(\Gamma_n, \alpha_n)$, ii) a sequence of histories $H_n$, iii) a new enumeration of pairs $(y_n, B_n)$, as follows:

- (step 0) Let $(\Gamma_0, \alpha_0) = (\Gamma, \alpha)$, $H_0 = H$, $(y_0, B_0) = (x, A)$.
- (step n+1) Given $(y_n, B_n)$, if $y_n \in \mathit{Lab}(\Gamma_n)$ and $\Gamma_n, \alpha_n \vdash^? y_n : B_n, H_n$ fails then proceed according to (a) else to (b).

(a) if $B_n$ if atomic, then we set

$$H_{n+1} = H_n \cup \{(y_n, B_n)\},$$
$$(\Gamma_{n+1}, \alpha_{n+1}) = (\Gamma_n, \alpha_n),$$
$$(y_{n+1}, B_{n+1}) = (x_{k+1}, A_{k+1}),$$
where $k = \max_{t \leq n} \exists s_{\leq n} (y_s, B_s) = (x_t, A_t)$,

else let $B_n = C \Rightarrow D$, then we set

$$H_{n+1} = H_n,$$
$$(\Gamma_{n+1}, \alpha_{n+1}) = (\Gamma_n, \alpha_n) \oplus_{y_n} (x_m : C),$$
$$(y_{n+1}, B_{n+1}) = (x_m, D),$$
where $x_m = \min\{x_t \in \mathcal{A} \mid x_t \notin \mathit{Lab}(\Gamma_n) \cup \mathit{Lab}(H_n)\}$.

(b) We set

$$H_{n+1} = H_n,$$
$$(\Gamma_{n+1}, \alpha_{n+1}) = (\Gamma_n, \alpha_n),$$
$$(y_{n+1}, B_{n+1}) = (x_{k+1}, A_{k+1}),$$
where $k = \max\{t \leq n \mid \exists s_{\leq n} (y_s, B_s) = (x_t, A_t)\}$,

**Construction of the Canonical model**

We define an **S**-model as follows $M = (W, R, V)$, such that

- $W = \bigcup_n \mathit{Lab}(\Gamma_n)$;
- $xRy \equiv \exists n A^{\mathbf{S}}_{\alpha_n}(x, y)$,
- $V(x) = \{q \mid \exists n\ x \in \mathit{Lab}(\Gamma_n)\ \wedge\ \Gamma_n, \alpha_n\ \vdash^?\ x : q, H_n$ succeeds$\}$.

The relation $R$ as defined above has the same properties of $A^{\mathbf{S}}$, e.g. if **S=S4**, that is $A^{\mathbf{S}}$ is transitive and reflexive, then so is $R$ and the same happens in all other cases. We can then show that for all $x \in W$ and formulas $B$,

$$M, x \models B\ \Leftrightarrow\ \exists n\ x \in \mathit{Lab}(\Gamma_n) \wedge \Gamma_n, \alpha_n\ \vdash^?\ x : B, H_n \text{ succeeds.}$$

We are mow able to conclude the proof of the completeness theorem. Let $f(z) = z$, for every $z \in \mathit{Lab}(\Gamma_0)$, where $(\Gamma_0, \alpha_0) = (\Gamma, \alpha)$ is the original database. It is easy to see that $f$ is a realization of $(\Gamma, \alpha)$ in $M$:

if $A^{\mathbf{S}}_{\alpha}(u, v)$ then $A^{\mathbf{S}}_{\alpha_0}(u, v)$, hence $f(u)Rf(v)$.

If $u : C \in \Gamma = \Gamma_0$, then by identity and the previous fact we have $M, f(u) \models C$. On the other hand, we have $M, f(x) \not\models A$ and $M, f(y) \not\models r$ for every $(y, r) \in H$. This concludes the proof. ■

COROLLARY 3.17. *If $A \models_{\mathbf{S}} B$ holds, then $A\ \vdash^?\ x_0 : B, \emptyset$, succeeds in P(**S**). In particular, if $A$ is valid in the modal system* **S**, *then $\vdash^?\ x_0 : A, \emptyset$, succeeds in P(**S**).*

## 3 SIMPLIFICATION FOR SPECIFIC SYSTEMS

In this section we show that for most of the modal logics we have considered, the use of labelled databases is not necessary and we can simplify either the structure of databases, or the deduction rules. We point out that the main results we have obtained for the general formulation with labels (cut-admissibility, completeness, restriction of the restart rule etc.) can be proved directly for each simplified system.

If we want to check the validity of a formula $A$, we evaluate $A$ from a trivial database $\vdash^?\ x_0 : A, \emptyset$. Restricting our attention to computations from trivial databases, we observe that we can only generate databases which have the form of trees.

DEFINITION 3.18. A database $(\Delta, \alpha)$ is called a *tree-database* if the set of links $\alpha$ forms a tree.

Let $(\Delta, \alpha)$ be a tree database and $x \in \mathit{Lab}(\Delta)$, we define the subdatabase $Path(\Delta, \alpha, x)$ as the *list* of labelled formulas lying on the path from the root of $\alpha$, say $x_0$, up to $x$, that is: $Path(\Delta, \alpha, x) = (\Delta', \alpha')$, where:

$$\begin{aligned} \alpha' &= \{(x_0, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n) \mid x_n = x \\ &\qquad \text{and for } i = 1, \ldots, n, (x_{i-1}, x_i) \in \alpha\} \\ \Delta' &= \{y : A \in \Delta \mid y \in \mathit{Lab}(\alpha')\}. \end{aligned}$$

PROPOSITION 3.19. *If a query $Q$ occurs in any derivation from a trivial database, then $Q = \Delta, \alpha \vdash^? z : B, H$, where $(\Delta, \alpha)$ is a tree-database.*

From now on we restrict our consideration to tree-databases.

## *3.1 Simplification for* K*,* K4*,* S4*,* KT*: Databases as Lists*

We show that for systems **K**, **K4**, **S4**, **KT** the proof procedure can be simplified in the sense that: (i) the databases are lists of formulas, (ii) the restart rule is not needed.

THEOREM 3.20. *If $\Delta, \alpha \vdash^? x : A, \emptyset$ succeeds, then $Path(\Delta, \alpha, x) \vdash^? x : A, \emptyset$ succeeds without using restart.*

By virtue of this theorem we can reformulate the proof system for logics from **K** to **S4** as follows. A database is simply a list of formulas $A_1, \ldots, A_n$, which stands for the labelled database $(\{x_1 : A_1, \ldots, x_n : A_n\}, \alpha)$, where $\alpha = \{(x_1, x_2), \ldots (x_{n-1}, x_n)\}$. A query has the form:

$$A_1, \ldots, A_n \vdash^? B$$

which represents $\{x_1 : A_1, \ldots, x_n : A_n\}, \alpha \vdash^? x_n : B$. The history has been omitted since restart is not needed. Letting $\Delta = A_1, \ldots, A_n$, we reformulate the predicates $A^{\mathbf{S}}$ as relations between formulas within a database $A^{\mathbf{S}}(\Delta, A_i, A_j)$, in particular we can define:

$$\begin{array}{lll} A^{\mathbf{K}}(\Delta, A_i, A_j) & \equiv & i + 1 = j \\ A^{\mathbf{KT}}(\Delta, A_i, A_j) & \equiv & i = j \vee i + 1 = j \\ A^{\mathbf{K4}}(\Delta, A_i, A_j) & \equiv & i < j \\ A^{\mathbf{S4}}(\Delta, A_i, A_j) & \equiv & i \leq j \end{array}$$

The rules become:

- (success) $\Delta \vdash^? q$ succeeds if $\Delta = A_1, \ldots, A_n$, and $A_n = q$;
- (implication) from

  $$\Delta \vdash^? A \Rightarrow B$$

  step to

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

- (reduction) from

  $$\Delta \vdash^? q$$

  step to

  $\Delta_i \vdash^? D_i$, for $i = 1, \ldots, k$, if there is a formula $A_j = D_1 \Rightarrow \ldots \Rightarrow D_k \Rightarrow q \in \Delta$, and there are integers $j = j_0 \leq j_1 \leq \ldots \leq j_k = n$, such that

$$i = 1, \ldots, k, A^{\mathbf{S}}(\Delta, A_{j_{i-1}}, A_{j_i}) \text{ holds and } \Delta_i = A_1, \ldots, A_{j_i}.$$

EXAMPLE 3.21. We show that $((b \Rightarrow a) \Rightarrow b) \Rightarrow c \Rightarrow (b \Rightarrow a) \Rightarrow a$ is a theorem of **S4**.

$$\begin{array}{rcll}
 & \vdash^? & ((b \Rightarrow a) \Rightarrow b) \Rightarrow c \Rightarrow (b \Rightarrow a) \Rightarrow a & \\
(b \Rightarrow a) \Rightarrow b & \vdash^? & c \Rightarrow (b \Rightarrow a) \Rightarrow a & \\
(b \Rightarrow a) \Rightarrow b, c & \vdash^? & (b \Rightarrow a) \Rightarrow a & \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a & \vdash^? & a & \text{reduction w.r.t. } b \Rightarrow a \text{ (1)} \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a & \vdash^? & b & \text{reduction w.r.t. } (b \Rightarrow a) \Rightarrow b \text{ (2)} \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a & \vdash^? & b \Rightarrow a & \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a, b & \vdash^? & a & \text{reduction w.r.t. } b \Rightarrow a \\
(b \Rightarrow a) \Rightarrow b, c, b \Rightarrow a, b & \vdash^? & b. &
\end{array}$$

This formula fails in both **KT** and **K4**, and therefore also fails in **K**: reduction at step (1) is allowed in **KT** but not in **K4**; on the contrary, reduction at step (2) is allowed in **K4** but not in **KT**.

## *3.2 Simplification for* **K5***,* **K45***,* **S5***: Databases as Clusters*

In this section we give an unlabelled formulation of logics **K5**, **K45**, **S5**. The simplification is allowed by the fact that we can define explicitly the accessibility relation.

PROPOSITION 3.22. *Let $Q = \Delta, \alpha \vdash^? x : G, H$ be any query which occurs in a P(**K5**) deduction from a trivial database $x_0 : A \vdash^? x_0 : B, H_0$. Let $R^{\mathbf{K5}}_\alpha(x, y)$ be defined as follows:*

$$\begin{aligned} R^{\mathbf{K5}}_\alpha(x, y) \quad \equiv \quad & (x = x_0 \wedge (x_0, y) \in \alpha) \\ & \vee \ (x, y \in \mathrm{Lab}(\alpha) \wedge x \neq x_0 \wedge y \neq x_0). \end{aligned}$$

*Then we have $R^{\mathbf{K5}}_\alpha(x, y) \equiv A^{\mathbf{K5}}_\alpha(x, y)$.*

COROLLARY 3.23. *Under the same conditions as the last proposition, we have:*

$$R^{\mathbf{K5}}_\alpha(x_0, x) \textit{ and } R^{\mathbf{K5}}_\alpha(x_0, y) \textit{ implies } x = y.$$

PROPOSITION 3.24. *Let $Q = \Delta, \alpha \vdash^? x : G, H$ be any query which occurs in a P(**K45**) deduction from a trivial database $x_0 : A \vdash^? x_0 : B, H_0$. Let $R^{\mathbf{K45}}_\alpha(x, y)$ be defined as follows:*

$$R^{\mathbf{K45}}_\alpha(x, y) \equiv x, y \in \mathrm{Lab}(\alpha) \wedge y \neq x_0.$$

*Then we have* $R_\alpha^{\mathbf{K45}}(x,y) \equiv A_\alpha^{\mathbf{K45}}(x,y)$.

PROPOSITION 3.25. *Let* $Q = \Delta, \alpha \vdash^? x : G, H$ *be any query which occurs in a P(**S5**) deduction from a trivial database* $x_0 : A \vdash^? x_0 : B, H_0$. *Let* $R_\alpha^{\mathbf{S5}}(x,y)$ *be defined as follows:*

$$R_\alpha^{\mathbf{S5}}(x,y) \equiv x, y \in \mathrm{Lab}(\alpha).$$

*Then we have* $R_\alpha^{\mathbf{S5}}(x,y) \equiv A_\alpha^{\mathbf{S5}}(x,y)$.

**Proof.** We observe that $A^{\mathbf{S5}}$ is the equivalence relation which contains just one class, namely the class of the labels which are descendants of $x_0$, that is all labels occurring in $\alpha$. ■

From the previous propositions we can reformulate the proof systems for **K5**, **K45** and **S5** without making use of labels. For **K5** the picture is as follows: either a database contains just one point $x_0$, or there is an initial point $x_0$ which is connected to another point $x_1$, and any point excluding $x_0$ is connected to any other. In the case of **K45**, $x_0$ is connected also to any point other than itself. Thus, in order to get a concrete structure without labels we must keep distinct the initial world from all the others, and we must indicate what is the current world, that is the world in which the goal formula is evaluated. In case of **K5** we must also identify the (only) world to which the initial world is connected. We are thus lead to consider the following structure.

A non-empty database has the form:

$$\Delta = B_0 \mid \ \mid \text{ or } \Delta = B_0 \mid B_1, \ldots, B_n \mid B_i, \text{ where } 1 \leq i \leq n,$$

and $B_0, B_1, \ldots, B_n$ are formulas. We also define

$$Actual(\Delta) = \begin{cases} B_0 \text{ if } \Delta = B_0 \mid \ \mid, \\ B_i \text{ if } \Delta = B_0 \mid B_1, \ldots, B_n \mid B_i. \end{cases}$$

This rather odd structure is forced by the fact that in **K5** and **K45** we have reflexivity in all worlds, except in the initial one and therefore, in contrast to all other systems, we have considered so far, the success of

> $\vdash^? x_0 : A \Rightarrow B$, which means that $A \Rightarrow B$ is valid,

does not imply the success of

> $x_0 : A \vdash^? x_0 : B$, which means that $A \to B$ is valid (material implication). [5]

[5]In these two systems the validity of $2C$ does not imply the validity of $C$, as it holds for all the other systems considered in this chapter.

The addition operation is defined as follows:

$$\Delta \oplus A = \begin{cases} B_0 \mid B_1, \ldots, B_n, A \mid A & \text{if } \Delta = B_0 \mid B_1, \ldots, B_n \mid B_i \\ B_0 \mid A \mid A & \text{if } \Delta = B_0 \mid \;\; \mid \\ \top \mid A \mid A & \text{if } \Delta = \emptyset \end{cases}$$

A query has the form

$\Delta \vdash^? G, H$, where $H = \{(A_1, q_1), \ldots, (A_k, q_k)\}$, with $A_j \in \Delta$.

DEFINITION 3.26 (Deduction Rules for **K5** and **K45**).
Given $\Delta = B_0 \mid B_1, \ldots, B_n \mid B$, let

$$\begin{array}{lcl} A^{\mathbf{K5}}(\Delta, X, Y) & \equiv & (X = B_0 \wedge Y = B_1) \\ & & \vee (X = B_i \wedge Y = B_j \wedge i, j > 0) \text{ and} \\ A^{\mathbf{K45}}(\Delta, X, Y) & \equiv & (X = B_i \wedge Y = B_j \text{ with } j > 0) \end{array}$$

- (success) $\Delta \vdash^? q, H$ succeeds if $Actual(\Delta) = q$.
- (implication) From $\Delta \vdash^? A \Rightarrow B, H$ step to $\Delta \oplus A \vdash^? B, H$.
- (reduction) if $\Delta = B_0 \mid B_1, \ldots, B_n \mid B$ and $C = D_1 \Rightarrow \ldots \Rightarrow D_k \Rightarrow q \in \Delta$, from $\Delta \vdash^? G, H$ step to

  $$B_0 \mid B_1, \ldots, B_n \mid C_i \vdash^? D_i, H \cup \{(B, q)\} \text{ for } i = 1, \ldots, k,$$

  for some $C_0, \ldots, C_k \in \Delta$, such that $C_0 = C$, $C_k = B_n$, and $A^{\mathbf{K5}}(\Delta, C_{i-1}, C_i)$ (respectively $A^{\mathbf{K45}}(\Delta, C_{i-1}, C_i)$) holds.
- (restart) If $\Delta = B_0 \mid B_1, \ldots, B_n \mid B_i$ and $(B_j, r) \in H$, with $j > 0$, then from $\Delta \vdash^? q, H$, step to

  $$B_0 \mid B_1, \ldots, B_n \mid B_j \vdash^? r, H \cup \{(B_i, q)\},$$

According to the above discussion, we observe that the check of the validity of $\models A \Rightarrow B$, corresponds to the query

$$\emptyset \vdash^? A \Rightarrow B, \emptyset,$$

which (by the implication rule) is reduced to the query

$$\top \mid A \mid A \vdash^? B, \emptyset.$$

This is different from checking the validity of $A \rightarrow B$ ($\rightarrow$ is the material implication), which corresponds to the query

$$A \mid \;\; \mid \vdash^? B, \emptyset.$$

The success of the former query does not imply the success of the latter. For instance in **K5**,

$\not\models (\top \Rightarrow p) \rightarrow p$ and indeed $\top \Rightarrow p \mid \quad \mid \vdash^? \; p, \emptyset$ fails.

On the other hand we have

$\models (\top \Rightarrow p) \Rightarrow p$ and indeed $\top \mid \top \Rightarrow p \mid \top \Rightarrow p \;\vdash^? \; p, \emptyset$ succeeds.

The reformulation of the proof system for **S5** is similar, but simpler. In the case of **S5**, there is no need to keep the first formula/world apart from the others. Thus, we may simply define a non-empty database as a pair $\Delta = (S, A)$, where $S$ is a set of formulas and $A \in S$. If $\Delta = (S, A)$, we let

$Actual(\Delta) = A$ and $\Delta \oplus B = (S \cup \{B\}, B)$.

For $\Delta = \emptyset$, we define $\emptyset \oplus A = (\{A\}, A)$. With these definitions the rules are similar to those of **K5** and **K45**, with the following simplifications:

- (reduction) if $\Delta = (S, B)$ and $C = D_1 \Rightarrow \ldots \Rightarrow D_k \Rightarrow q \in \Delta$, then from $\Delta \;\vdash^? \; G, H$ step to

  $(S, C_i) \;\vdash^? \; D_i, H \cup \{(B, q)\}$, where for $i = 1, \ldots, k$ $C_i \in \Delta$ and $C_k = B$.

- (restart) If $\Delta = (S, B)$ and $(C, r) \in H$, then from $(S, B) \;\vdash^? \; q, H$, step to

  $(S, C) \;\vdash^? \; r, H \cup \{(B, q)\}$,

EXAMPLE 3.27. In Figure 3.3 we show a derivation of the following formula in **S5**

$((a \Rightarrow b) \Rightarrow c) \Rightarrow (a \Rightarrow d \Rightarrow c) \Rightarrow (d \Rightarrow c)$.

In the derivation we make use of restricted restart, according to Proposition 3.6. A brief explanation of the derivation: step (5) is obtained by reduction w.r.t. $(a \Rightarrow b) \Rightarrow c$, step (7) by restart, steps (8) and (9) by reduction w.r.t. $a \Rightarrow d \Rightarrow c$, and they both succeed immediately.

## 4 EXTENDING THE LANGUAGE

In this section we extend the proof procedures into a broader language. We do our extension in three steps. We first consider a simple extension allowing conjunction, then we consider a richer fragment containing 'local clauses' and $(\wedge, \vee)$-combinations of goals. This fragment can be considered as an analog of Harrop formulas in intuitionistic logic. We finally extend the procedures to the full propositional modal language via a translation into an implicative normal form.

$$(1)\quad \vdash^{?} \ ((a \Rightarrow b) \Rightarrow c) \Rightarrow (a \Rightarrow d \Rightarrow c) \Rightarrow d \Rightarrow c$$

$$(2)\ \{(a \Rightarrow b) \Rightarrow c\}, (a \Rightarrow b) \Rightarrow c \ \vdash^{?} \ (a \Rightarrow d \Rightarrow c) \Rightarrow d \Rightarrow c$$

$$(3)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c\}, a \Rightarrow d \Rightarrow c \ \vdash^{?} \ d \Rightarrow c$$

$$(4)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d\}, d \ \vdash^{?} \ c$$

$$(5)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d\}, d \ \vdash^{?} \ a \Rightarrow b, (\mathbf{d}, \mathbf{c})$$

$$(6)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d, a\}, a \ \vdash^{?} \ b, (\mathbf{d}, \mathbf{c})$$

$$(7)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d, a\}, d \ \vdash^{?} \ c, (\mathbf{d}, \mathbf{c})$$

$$(8)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d, a\}, a \ \vdash^{?} \ a, (\mathbf{d}, \mathbf{c}) \qquad (9)\ \{(a \Rightarrow b) \Rightarrow c, a \Rightarrow d \Rightarrow c, d, a\}, d \ \vdash^{?} \ d, (\mathbf{d}, \mathbf{c})$$

Figure 3.3. Derivation for Example 3.27.

## *Conjunction*

To handle conjunction in the labelled formulation, we simply drop the condition that a label $x$ may be attached to only one formula, thus formulas with the same label can be thought as logically conjuncted. In the unlabelled formulation, for those systems enjoying such a formulation, the general principle is to deal with *sets* of formulas, instead of *single formulas*. A database will be a structured collection of *sets* of formulas, rather than a collection of formulas. The structure is always the same, but the constituents are now sets. Thus, in the cases of **K**, **KT**, **K4** and **S4**, databases will be lists of *sets* of formulas, whereas in the cases of **K5**, **K45** and **S5**, they will be clusters of *sets* of formulas.

A conjunction of formulas is interpreted as a set, so that queries may contain *sets* of goal formulas.

A formula $A$ of language $\mathcal{L}(\wedge, \Rightarrow)$ is in *normal form* if it is an atom or has form:

$$\bigwedge_i [S^i_1 \Rightarrow \ldots \Rightarrow S^i_{n_i} \Rightarrow q_i]$$

where $S^i_j$ are conjunctions of formulas in *normal form*. In all modal logics considered in this chapter (formulated $\mathcal{L}(\wedge, \Rightarrow)$) it holds that every formula has an equivalent one in normal form.

We simplify the notation for NF formulas and replace conjunctions with sets. For example the NF of $(b \Rightarrow (c \wedge d)) \Rightarrow (e \wedge f)) \wedge ((g \wedge h) \Rightarrow (k \wedge u))$ is the set containing the following formulas:

$$\{b \Rightarrow c, b \Rightarrow d\} \Rightarrow e,\ \{b \Rightarrow c, b \Rightarrow d\} \Rightarrow f,\ \{g, h\} \Rightarrow k,\ \{g, h\} \Rightarrow u.$$

For the deduction procedures all we have to do is to handle sets of formulas. We define, for $x \in Lab(\Gamma) \cup Lab(H)$, $y \notin Lab(\Gamma) \cup Lab(H)$ and finite set of formulas $S = \{D_1, \ldots, D_t\}$,

$$(\Gamma, \alpha) \oplus_x y : S = (\Delta \cup \{y : D_1, \ldots, y : D_t\}, \alpha \cup \{(x, y)\}),$$

then we change the *(implication) rule* in the obvious way:

from $\Delta, \alpha \ \vdash^? \ x : S \Rightarrow B, H$,

step to

$(\Delta, \alpha) \oplus_x y : S \ \vdash^? \ y : B, H$,

where $S$ is a set of formulas in NF and $y \notin Lab(\Gamma)$, and we add a rule for proving sets of formulas:

from $(\Delta, \alpha) \ \vdash^? \ x : \{B_1, \ldots B_k\}, H$

step to

$(\Delta, \alpha) \ \vdash^? \ x : B_i, H$ for $i = 1, \ldots, k$.

Regarding the simplified formulations without labels, the structural restrictions in the rules (reduction and success) are applied to the sets of formulas, which are now the constituents of databases, considered as units; the history $H$, when needed, becomes a set of pairs $(S_i, A_i)$, where $S_i$ is a set and $A_i$ is a formula. The property of restricted restart still holds for this formulation.

EXAMPLE 3.28. We show that the following is a theorem of **K5**, using the unlabelled formulation:

$$\Box(b \to c) \wedge \Box(\Box(\Box(a \to b) \to d) \to c) \to \Box(a \to c).$$

Letting $S_0 = \{b \Rightarrow c, ((a \Rightarrow b) \Rightarrow d) \Rightarrow c\}$, the initial query is

$S_0 \mid \mid \vdash^? \; a \Rightarrow c.$

In the derivation below, we make use of restricted restart.

$$\begin{array}{rcl}
S_0 \mid \mid & \vdash^? & a \Rightarrow c \\
S_0 \mid \{a\} \mid \{a\} & \vdash^? & c, \; (\{a\}, c) \\
S_0 \mid \{a\} \mid \{a\} & \vdash^? & (a \Rightarrow b) \Rightarrow d, (\{a\}, c) \text{ by reduction w.r.t.} \\
 & & \quad ((a \Rightarrow b) \Rightarrow d) \Rightarrow c \in S_0 \\
S_0 \mid \{a\}, \{a \Rightarrow b\} \mid \{a \Rightarrow b\} & \vdash^? & d, \; (\{a\}, c) \\
S_0 \mid \{a\}, \{a \Rightarrow b\} \mid \{a\} & \vdash^? & c, \; (\{a\}, c) \; \text{ by restart} \\
S_0 \mid \{a\}, \{a \Rightarrow b\} \mid \{a\} & \vdash^? & b, \; (\{a\}, c) \; \text{ by reduction w.r.t. } b \Rightarrow c \in S_0 \\
S_0 \mid \{a\}, \{a \Rightarrow b\} \mid \{a\} & \vdash^? & a, \; (\{a\}, c) \; \text{ by reduction w.r.t. } a \Rightarrow b \\
 & & \text{success.}
\end{array}$$

### *Modal Harrop Formulas*

We can further extend the language in a similar way to [Giordano and Martelli, 1994], by allowing *local clauses* of the form

$G \to q,$

where $\to$ denotes ordinary (material) implication. We call them 'local', since $x : G \to q$ can be used only in world $x$ to reduce the goal $x : q$ and it is not usable/visible in any other world. It is 'private' to $x$. The extension we suggest may be significant in developing logic programming languages based on modal logic [Giordano *et al.*, 1992; Giordano and Martelli, 1994]. To this end, we can introduce a sort of modal Harrop fragment distinguishing database (D-formulas) and goal formulas (G-formulas).

Below we define D-formulas i.e. the ones which may occur in the database and G-formulas i.e. the one which may occur in goal position. The former are further distinct in *modal* D-formulas (MD) and *local* D-formulas (LD).

$LD := G \to q,$
$MD := \top \mid q \mid G \Rightarrow MD,$
$D := LD \mid MD,$
$CD := D \mid CD \wedge CD;$
$G := \top \mid q \mid G \wedge G \mid G \vee G \mid CD \Rightarrow G.$

We also use $\Box G$ and $\Box D$ as syntactic sugar for $\top \Rightarrow G$ and $\top \Rightarrow D$. Notice that atoms are both LD- and MD-formulas (as $\top \to q \equiv q$); moreover, any non-atomic MD-formula can be written as $G_1 \Rightarrow \ldots \Rightarrow G_k \Rightarrow q$. Finally, CD formulas are just conjunction of D-formulas.

For D- and G-formulas as defined above we can easily extend the proof procedure. We give it in the most general formulation for labelled databases. It is clear that one can derive an unlabelled formulation for systems which allow it, as explained in the previous section. In the labelled formulation, queries have the form

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

where $\Delta$ is a set of D-formulas, $G$ is a G-formula, and $H = \{(x_1, G_1), \ldots, (x_k, G_k)\}$, where $G_i$ are G-formulas. The **additional** rules are:

- (true) $\Delta, \alpha \vdash^? x : \top, H$ immediately succeeds.
- (local-reduction) From $\Delta, \alpha \vdash^? x : q, H$
  step to

  $$\Delta, \alpha \vdash^? x : G, H \cup \{(x : q)\}$$

  if $x : G \to q \in \Delta$.
- (and) From $\Delta, \alpha \vdash^? x : G_1 \wedge G_2, H$
  step to

  $$\Delta, \alpha \vdash^? x : G_1, H \text{ and } \Delta, \alpha \vdash^? x : G_2, H.$$
- (or) From $\Delta, \alpha \vdash^? x : G_1 \vee G_2, H$
  step

  to $\Delta, \alpha \vdash^? x : G_1, H \cup \{(x, G_2)\}$ or to $\Delta, \alpha \vdash^? x : G_2, H \cup \{(x, G_1)\}$.

EXAMPLE 3.29. Let $\Delta$ be the following database

$x_0 : [(\Box p \Rightarrow s) \wedge b] \to q,$
$x_0 : ([(p \Rightarrow q) \wedge \Box a] \Rightarrow r) \to q,$
$x_0 : a \to b.$

We show that $\Delta, \emptyset \vdash^? x_0 : q, \emptyset$ succeeds in the proof system for **KB** and this shows that the formula

$$
\begin{array}{c}
(1)\ \Delta \vdash^? x_0 : q, \emptyset \\
| \\
(2)\ \vdash^? x_0 : [(p \Rightarrow q) \wedge \Box a] \Rightarrow r, (x_0, q) \\
| \\
(3)\ x_1 : p \Rightarrow q, x_1 : \Box a, \{(x_0, x_1)\} \vdash^? x_1 : r, (x_0, q) \\
| \\
(4)\ \vdash^? x_0 : q, (x_0, q)
\end{array}
$$

$$
\begin{array}{cc}
(5)\ \vdash^? x_0 : \Box p \Rightarrow s, (x_0, q) & (10)\ \vdash^? x_0 : b, (x_0, q) \\
| & | \\
(6)\ x_2 : \Box p, \{(x_0, x_1), (x_0, x_2)\} \vdash^? x_2 : s, (x_0, q) & (11)\ \vdash^? x_0 : a, (x_0, q) \\
| & | \\
(7)\ \vdash^? x_0 : q, (x_0, q) & (12)\ \vdash^? x_0 : \top, (x_0, q) \\
| & \\
(8)\ \vdash^? x_0 : p, (x_0, q) & \\
| & \\
(9)\ \vdash^? x_0 : \top, (x_0, q) &
\end{array}
$$

Figure 3.4. Derivation for Example 3.29.

$$\bigwedge \Delta \to q \text{ is valid in } \mathbf{KB}.$$

A derivation is shown in Figure 3.4. The property of restricted restart still holds for this fragment, thus we do not need to record the entire history, but only the first pair $(x_0, q)$. At each step we only show the additional data introduced in that step. A quick explanation of the steps: step (2) is obtained by local reduction w.r.t. $x_0 : ([(p \Rightarrow q) \wedge \Box a] \Rightarrow r) \to q$, step (4) by restart, steps (5) and (10) by local reduction w.r.t. $x_0 : [(\Box p \Rightarrow s) \wedge b] \to q$, step (7) by restart, step (8) by reduction w.r.t. $x_1 : p \Rightarrow q$ since letting $\alpha = \{(x_0, x_1), (x_0, x_2)\}$ $A^{\mathbf{KB}}_{\alpha}(x_1, x_0)$ holds; step (9) is obtained by reduction w.r.t. $x_2 : \Box p\ (= \top \Rightarrow p)$ since $R^{\mathbf{KB}}_{\alpha}(x_2, x_0)$ holds, step (11) by local reduction w.r.t. $x_0 : a \to b$, step (12) by reduction w.r.t. $x_1 : \Box a\ (= \top \Rightarrow a)$ since $R^{\mathbf{KB}}_{\alpha}(x_1, x_0)$ holds.

The soundness and completeness results of Section 2.1 can be extended to this fragment; to this regard let the validity of a query be defined as in that section.

THEOREM 3.30. $\Delta \vdash^? G, H$ *succeeds in P(***S***) if and only if it is valid.*

## *Extension to the Whole Propositional Language*

In this section we extend the procedure to the whole propositional modal language. The idea is very simple: we take the computation procedure for the classical logic of Chapter 2 and we combine it with the modal procedure of this chapter. Before presenting it, let us see what the difficulty is of such an extension. Clearly the trouble is the treatment of the $\Diamond$ modality, i,e. the possibility operator, which is dual to $\Box$. For a goal-directed procedure, the problem is similar to admitting data of the form $\exists x B$ in a first-order database. Intuitively, a goal

$$\vdash^? \; x : \Diamond A$$

should succeed if $\vdash^? \; y : A$ succeeds for some $y$ accessible from $x$. The point is that such $y$ may not yet exist when $x : \Diamond A$ is asked, whence the goal $x : \Diamond A$ has to be properly delayed. However, taking advantage of the history component of a query, we can think of implementing this rule directly. What is real trouble is the treatment of the $\Diamond$-formulas in the database:

$$x : \Diamond A \; \vdash^? \; y : G.$$

What might be the contribution of $x : \Diamond A$ to the proof of $y : G$? In this case, we can think of treating $x : \Diamond A$ as soon as it enters the database by immediately creating an $x$-accessible $z$ with $z : A$. This would correspond to a sort of *Skolemization* on worlds, in analogy with the handling of existential data in a first-order language. But what to do with

$$A \to \Diamond B \text{ or } A \Rightarrow \Diamond B, \text{ i.e. } \Box(A \to \Diamond B) \; ?$$

Remember that we want to stick to the goal-directed paradigm. In the procedure, we give below we simply remove the problem by translating $\Diamond A$ into $\Box(A \to \bot) \to \bot$, or equivalently into $(A \Rightarrow \bot) \to \bot$. However, it will turn out that the implicit treatment of the possibility operator does conform with the discussion above.

To minimize the work, we introduce a simple normal form on the set of connectives $(\Rightarrow, \to, \wedge, \top, \bot)$. It is obvious that this set forms a complete base. The normal form is an immediate extension of the normal form for $\Rightarrow, \wedge$.

PROPOSITION 3.31. *Every modal formula over the language $(\to, \neg, \Diamond, \Box)$ is equivalent to a set (conjunction) of NF-formulas of the form*

$$S_0 \to (S_1 \Rightarrow (S_2 \Rightarrow \ldots \Rightarrow (S_n \Rightarrow q) \ldots)$$

*where $q$ is an atom, $\top$, or $\bot$, $n \geq 0$, and each $S_i$ is a conjunction (set) of NF-formulas.*

As usual we omit parentheses, so that the above will be written as

$$S_0 \to S_1 \Rightarrow S_2 \Rightarrow \ldots \Rightarrow S_n \Rightarrow q.$$

In practice we will replace the conjunction by the set notation as we wish. When we need it, we distinguish two types of NF-formulas, (i) those with non-empty $S_0$, which are written as above, and (ii) those with empty $S_0$, which are simplified to $S_1 \Rightarrow S_2 \Rightarrow \ldots \Rightarrow S_n \Rightarrow q$. For a quick case analysis, we can also say that type (i) formulas have the form $S \rightarrow D$, and type (ii) have the form $S \Rightarrow D$, where $D$ is always of type (ii).

For instance, the NF-form of $p \rightarrow \Diamond r$ is

$$(p \wedge (r \Rightarrow \bot)) \rightarrow \bot, \text{ or equivalently } \{p, r \Rightarrow \bot\} \rightarrow \bot.$$

This formula has the structure $S_0 \rightarrow q$, where $S_0 = \{p, r \Rightarrow \bot\}$ and $q = \bot$. The NF-form of

$$\Box(\Diamond a \rightarrow \Diamond(b \wedge c))$$

is given by

$$((a \Rightarrow \bot) \rightarrow \bot) \Rightarrow ((b \wedge c) \Rightarrow \bot) \Rightarrow \bot.$$

We give below the rules for queries of the form

$$\Gamma, \alpha \vdash^? x : G, H,$$

where $\Gamma$ is a labelled set of NF-formulas, $G$ is a NF-formula, $\alpha$ is a set of links (as usual), $H$ is a set of pairs $\{(x_1, q_1), \ldots (x_k, q_k)\}$, where $q_i$ is an atom.

DEFINITION 3.32 (Deduction rules for whole modal logics). For each modal system **S**, the corresponding proof system, denoted by P(**S**), comprises the following rules, parametrized to predicates $A^{\mathbf{S}}$.

- (success) $\Delta, \alpha \vdash^? x : q, H$ immediately succeeds if $q$ is an atom and $x : q \in \Delta$.
- (strict implication) From $\Delta, \alpha \vdash^? x : S \Rightarrow D, H$, step to

  $$(\Delta, \alpha) \oplus_x (y : S) \vdash^? y : D, H,$$

  where $y \notin \mathit{Lab}(\Delta) \cup \mathit{Lab}(H)$.
- (implication) From $\Delta, \alpha \vdash^? x : S \rightarrow D, H$, step to

  $$\Delta \cup \{x : A \mid A \in S\}, \alpha \vdash^? x : D, H.$$
- (reduction) If $y : C \in \Delta$, with $C = S_0 \rightarrow S_1 \Rightarrow S_2 \Rightarrow \ldots \Rightarrow S_k \Rightarrow q$, with $q$ atomic, then from

  $$\Delta, \alpha \vdash^? x : q, H$$

  step to

  $$\Delta, \alpha \vdash^? u_0 : S_0, H' \quad \Delta, \alpha \vdash^? u_1 : S_1, H' \quad \ldots \quad \Delta, \alpha \vdash^? u_k : S_k, H'$$

  where $H' = H$ if $q = \bot$, and $H' = H \cup \{(x, q)\}$ otherwise, for some $u_0, \ldots, u_k \in \mathit{Lab}(\alpha)$, such that $u_0 = y$, $u_k = x$, and

$A^{\mathbf{S}}_{\alpha}(u_i, u_{i+1})$ holds, for $i = 0, \ldots, k-1$.

- (restart) If $(y, r) \in H$, then, from $\Delta, \alpha \vdash^? x : q, H$, with $q$ atomic, step to

  $\Delta, \alpha \vdash^? y : r, H \cup \{(x, q)\}$.

- (falsity) From $\Delta, \alpha \vdash^? x : q, H$, if $y \in Lab(\Gamma)$ step to

  $\Delta, \alpha \vdash^? y : \bot, H \cup \{(x : q)\}$.

- (conjunction) From

  $(\Delta, \alpha) \vdash^? x : \{B_1, \ldots B_k\}, H$

  step to

  $(\Delta, \alpha) \vdash^? x : B_i, H$ for $i = 1, \ldots, k$.

If the number of subgoals is 0, i.e. $k = 0$, the above reduction rule becomes the rule for *local clauses* of the previous section. On the other hand if $S_0 = \emptyset$, then the query with goal $S_0$ is omitted and we have the rule of Section 2.

This procedure is actually a minor extension of the one based on strict-implication/conjunction. The rule for falsity may be source of non-determinism, as it can be applied to any label $y$. Further investigation should clarify to what extent this rule is needed and if it is possible to restrict its applications to special cases. However, the proof procedure is sound and complete, as asserted in the next theorem.

THEOREM 3.33. $\Gamma, \gamma \vdash^? x : G, H$ *succeeds if and only if it is valid.*

EXAMPLE 3.34. In **K5** we have $\Diamond p \to \Box\Diamond p$. This is translated as $((p \Rightarrow \bot) \to \bot) \to ((p \Rightarrow \bot) \Rightarrow \bot)$. Below we show a derivation. Some explanation and remarks: at step (1) we can only apply the rule for falsity, or reduction w.r.t. $y : p \Rightarrow \bot$, since $A^{\mathbf{K5}}_{\alpha}(x_0, y)$ implies $A^{\mathbf{K5}}_{\alpha}(y, y)$. We apply the rule for falsity. The reduction at step (2) is legitimate as $A^{\mathbf{K5}}_{\alpha}(x_0, y)$ and $A^{\mathbf{K5}}_{\alpha}(x_0, z)$ implies $A^{\mathbf{K5}}_{\alpha}(y.z)$.

| | | |
|---|---|---|
| | $\vdash^? x_0 : ((p \Rightarrow \bot) \to \bot) \to ((p \Rightarrow \bot) \Rightarrow \bot$ | |
| $x_0 : (p \Rightarrow \bot) \to \bot$ | $\vdash^? x_0 : (p \Rightarrow \bot) \Rightarrow \bot$ | |
| $y : p \Rightarrow \bot,\ \alpha = \{(x_0, y)\}$ | $\vdash^? y : \bot$ | (1) |
| | $\vdash^? x_0 : \bot$ rule for $\bot$ | |
| | $\vdash^? x_0 : p \Rightarrow \bot$ | |
| $z : p,\ \alpha = \{((x_0, y), (x_0, z)\}$ | $\vdash^? z : \bot$ | (2) |
| | $\vdash^? z : p$ | |
| | success | |

EXAMPLE 3.35. In **K** we have

$$[\Box\Box(B \to C) \wedge \Box(A \to \Diamond(B \vee C)) \wedge \Diamond A] \to \Diamond\Diamond C].$$

Translating this formula into our language, we have to show that $\Delta, \emptyset \vdash^? x : G$ succeeds, where $\Delta$ is:

$$\begin{array}{l} x : \top \Rightarrow B \Rightarrow C, \\ x : \{A, ((B \to C) \to C) \Rightarrow \bot\} \Rightarrow \bot, \\ x : (A \Rightarrow \bot) \to \bot, \end{array}$$

and $G$ is $(((C \Rightarrow \bot) \to \bot) \Rightarrow \bot) \to \bot$. That is we must show:

$$\Delta \cup \{x : ((C \Rightarrow \bot) \to \bot) \Rightarrow \bot\}, \emptyset \vdash^? x : \bot,\ \emptyset \text{ succeeds.}$$

We assume that $A, B, C$ are atoms. In Figure 3.5 we show a derivation. At each step we only show the new data inserted in the database at that step (if any). We also omit the history which is uniquely determined by the branch. Step (1) is obtained by reduction w.r.t. $x : (A \Rightarrow \bot) \to \bot$; notice that no other formula can be used for reduction at this point. Step (3) is obtained by reduction w.r.t. $x : ((C \Rightarrow \bot) \to \bot) \Rightarrow \bot$. Steps (5) and (6) are obtained by reduction w.r.t. $x : (\{A, ((B \to C) \to C) \Rightarrow \bot\} \Rightarrow \bot$. Step (8) is obtained by reduction w.r.t. $y : C \Rightarrow \bot$ introduced in (4). Step (9) is obtained by reduction w.r.t. $z : (B \to C) \to C$, and steps (11) and (12) by reduction w.r.t. $x : \top \Rightarrow B \Rightarrow C$.

Examining the computation we can understand how a formula $\Diamond A$ is dealt. A goal $x : \Diamond A$ is translated into

$$\Delta, \alpha \vdash^? x : (A \Rightarrow \bot) \to \bot,$$

from which we step to

$$\Delta, x : A \Rightarrow \bot, \alpha \vdash^? x : \bot.$$

This will succeed using the additional data if $y : A$ is provable for a world $y$, accessible from $x$, (i.e. such that $A^{\mathbf{S}}_{\alpha}(x, y)$ hold).This is exactly what we expect from an evaluation of a goal $\Diamond A$. On the other hand having $x : \Diamond A$ in the database corresponds to having $x : (A \Rightarrow \bot) \to \bot$. If this is used in any deduction, it will be used as follows: from

$$x : (A \Rightarrow \bot) \to \bot,\ \alpha \vdash^? x : \bot$$

we step to

$$x : (A \Rightarrow \bot) \to \bot, y : A,\ \alpha \cup \{(x, y)\} \vdash^? y : \bot,$$

that is we create a new world $y$ accessible from $x$ with $A$ holding in it. Again this is what we expect. Notice, however, that because of the unrestricted falsity rule such a step is enabled whenever any atomic goal is reached. Specific restrictions on the application of the falsity rule should be studied.

$$\Delta \cup \{x : (((C \Rightarrow \bot) \to \bot) \Rightarrow \bot\}, \emptyset \ \vdash^? \ x : \bot$$

$$(1) \ \vdash^? \ x : A \Rightarrow \bot$$

$$(2) \ y : A, \{(x, y)\} \ \vdash^? \ y : \bot$$

$$(3) \ \vdash^? \ y : (C \Rightarrow \bot) \to \bot$$

$$(4) \ y : C \Rightarrow \bot \ \vdash^? \ y : \bot$$

$$(5) \ \vdash^? \ y : A \qquad\qquad (6) \ \vdash^? \ y : ((B \to C) \to C) \Rightarrow \bot$$

$$(7) \ z : (B \to C) \to C, \{(x, y), (y, z)\} \ \vdash^? \ z : \bot$$

$$(8) \ \vdash^? \ z : C$$

$$(9) \ \vdash^? \ z : B \to C$$

$$(10) \ z : B \ \vdash^? \ z : C$$

$$(11) \ \vdash^? \ y : \top \qquad\qquad (12) \ \vdash^? \ z : B$$

Figure 3.5. Derivation for Example 3.34.

A final remark. The extension of the goal-directed procedure to the full propositional modal logic leads us also to consider systems where the accessibility relation satisfies the *seriality property* i.e. $\forall x \exists y\ xRy$; the treatment of these systems were excluded because of the restriction to the strict implication fragment (see Footnote 2, p. 50). We think that the rule for seriality is the following, namely a liberalization of the falsity rule: from

$$\Delta, \alpha \ \vdash^{?} \ x : q, H$$

step to

$$\Delta, \alpha' \ \vdash^{?} \ y : \bot, H \cup \{(x, q)\}$$

where either $y \in \mathit{Lab}(\Delta)$ and $\alpha' = \alpha$, or $y \notin \mathit{Lab}(\Delta) \cup \mathit{Lab}(H)$ and $\alpha' = \alpha \cup \{(x, y)\}$. We *conjecture* that if we take this rule in place of the falsity rule, we obtain complete proof systems for modal logics with the axiom **D** i.e. $\Box A \rightarrow \Diamond A$ which corresponds to seriality. The reader is invited to check (an instance of) **D** where $A$ is an atom in the proof system $P(\mathbf{K})$ plus the rule above (its translation in NF is $\{\top \Rightarrow A, A \Rightarrow \bot\} \rightarrow \bot$).

# CHAPTER 4

# SUBSTRUCTURAL LOGICS

## 1 INTRODUCTION

In this chapter we consider substructural implicational logics. These logics, as far as the implicational fragment is concerned, put some restrictions on the use of formulas in deductions. The restrictions may require either that every formula of the database must be used, or that it cannot be used more than once, or that it must be used according to a given ordering of database formulas. The denomination *substructural logics* can be explained by the fact that these systems restrict (some of) the structural rules of deduction, that is to say the *contraction*, *weakening* and *permutation* rules which are allowed both by classical and intuitionistic provability. Substructural logics have received an increasing interest in the computer science community because of their potential applications in a number of different areas, such as natural language processing, database update, logic programming, type-theoretic analysis by means of the so called Curry–Howard isomorphism and, more generally, the categorical interpretation of logics. We refer to [Došen, 1993; Ono, 1998; Ono, 1993; Routley *et al.*, 1982] for a survey. Recently, Routley–Meyer semantics for substructural logics have been re-interpreted as modelling agent-interaction [Slaney and Meyer, 1997].

We give a brief survey of the implicational systems we consider in this chapter. The background motivation of *relevant logics* is the attempt to formalize an intuitive requirement on logical deductions: in any deduction the hypotheses must be relevant to the conclusion. *Relevant* means that the hypotheses must actually be used to get the conclusion. Thus the first intuition behind relevant logics is to reject the *weakening rule*

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

since $B$ may have nothing to do with the proof of $A$ from $\Gamma$. In the object language this rule is reflected by the theorem (of classical, likewise intuitionistic logic)

(Irrelevance) $A \to (B \to A)$.

The implicational fragment of Relevant logic **R** can be axiomatized by dropping Irrelevance from the axiomatization of intuitionistic implication. A sequent calculus for the implicational fragment of **R** can be obtained from the sequent formulation of (the implicational fragment of) intuitionistic logic by restricting the identity axiom to single formulas $A \vdash A$, and by dropping the above weakening rule. Whereas system **R** takes into account the basic concern about the use of formulas, other systems such as **E** and **T** put some further restrictions on the order in which formulas can be used in any deduction. These restrictions can be expressed in a sequent formulation as restrictions on the rule of *permutation* (see below), although in a less natural way.[1]

The system **E** combines relevance and necessity. The implication of **E** can be read at the same time as relevant implication and strict implication. If we think of interpreting necessary as *forced by truth*, we can try to define

$$\Box A =_{def} (A \to A) \to A.$$

With this interpretation the logic **R** cannot be interpreted as a logic of necessity, since in **R** we have both

$$A \to \Box A \text{ and } \Box A \to A.$$

Thus the modality collapses.

The logic **E** rejects the thesis $A \to \Box A$, avoiding the collapse of modality; the implication $\to$ of **E** can be read as strict implication as in **E** we have

$$A \to B \to \Box(A \to B) \quad \text{and} \quad \Box(A \to B) \to (A \to B).$$

The interpretation of modality is that one of **S4**, as in **E** we have:

$\Box(A \to B) \to \Box A \to \Box B$,
$\Box A \to A$,
$\Box A \to \Box\Box A$,
if $\vdash A$ then $\vdash \Box A$.

In the system **T** we cut the relationship between $\Box A$ and $A$, by rejecting the thesis $\Box A \to A$ as well. But the intuition behind **T** stems from a concern about the use of the two hypotheses in an inference by Modus Ponens: the restriction is that the minor $A$ must not be derived 'before' the ticket $A \to B$. This is clarified by the Fitch-style natural deduction of **T**, for which we refer to [Anderson and Belnap, 1975]; in terms of natural deduction, the requirement on Modus Ponens says that the minor must be derived either in the same context of the ticket (that is from the same assumptions) or in a nested subproof occurring in the proof of the ticket.

None of the systems mentioned put any bound on the number of times a formula can be used within a deduction, what only matters is that it is used at least once. On the contrary, a few of the systems we consider pay attention to the number of times a formula can be used in a deduction. Namely, in some logics, such as **BCK** and **L**, a formula cannot be used more than once. From the point of view of

[1] A sequent formulation of most relevant logics which does not make use of any additional machinary is the so- called 'merge' formulation presented in [Anderson and Belnap, 1975]. On the other hand, many authors, [Anderson *et al.*, 1992; Došen, 1988; Goré, 1998] have developed sequent calculi for substructural logics putting additional structure in sequents along the line of Belnap's Display logic.

a sequent formulation, this constraint about the use of a formula can be expressed by removing the rule of contraction

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

from the standard sequent system for intuitionistic logic. In this context, formulas may be thought as representing *resources* which are consumed through a logical deduction. This motivates an alternative denomination of substructural logics as *bounded resource logics* [Gabbay, 1996]. In more detail, **BCK** is the logic which result from intuitionistic logic by dropping contraction. **L** reject both weakening and contraction. The implicational logic **L** is the implicational fragment of linear logic [Girard, 1987].[2].

We will also consider contractionless versions of **E** and **T**, **E-W**and **T-W** respectively.

The weakest system we consider is **FL**[3], which is related to the *right implicational* fragment of Lambek calculus, proposed by Lambek in [1958] as a calculus of syntactic categories for natural language. This system rejects all substructural rules, in particular, in addition to weakening and contraction, it rejects the law of permutation:

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}.$$

We will mainly concentrate on the implicational fragment of the systems mentioned. In Section 3, we will extend the proof systems to a fragment similar to Harrop-formulas. For the fragment considered, all the logics studied in this chapter are subsystems of intuitionistic logic. However, this is no longer true for the fragment comprising an involutive negation, which can be added (and has been added) to each system. In this chapter we do not consider the treatment of negation. We refer the reader to [Anderson and Belnap, 1975; Anderson *et al.*, 1992] for an extensive discussion.

In Figure 1 we show the inclusion relation of the systems we consider in this chapter.

As we have seen in the previous chapters, we can control the use of formulas by labelling data and putting constraints on the labels. In this specific context by labelling data, we are able to record whether they have been used or not and to express the additional conditions needed for each specific systems. The use of labels to deal with substructural logics is not a novelty: it has been used for instance by Prawitz [1965], by Anderson and Belnap [1975], to develop a natural-deduction formulation of most relevant logics. Our proof systems are similar to

[2]Linear logic is acutally a much more articulated logic. It has an intuitionistic as well as a classical version. In linear logic the weakening and contraction rules are re-introduced in a controlled way by special operators called *exponentials*. What we call here **L** is also commonly known as **BCI** logic.

[3]The denomination of the system is taken from [Ono, 1998; Ono, 1993].

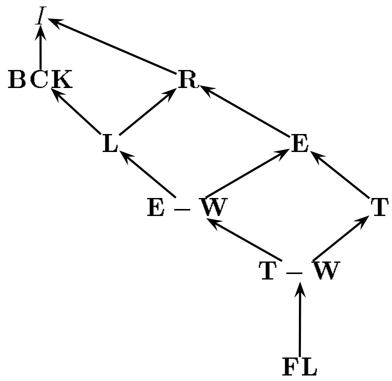


Figure 4.1. Lattice of Substructural Logics.

their natural deduction systems in this respect: we do not have explicit structural rules. The structural rules are *internalized* as restrictions in the logical rules.

## 2 PROOF SYSTEMS

In this section we develop proof methods for the implicational logics: **R**, **BCK**, **E**, **T**, **E-W**, **T-W**, **FL**.[4]

DEFINITION 4.1. Let us fix a denumerable alphabet $\mathcal{A} = \{x_1, \ldots, x_i, \ldots\}$ of labels. We assume that labels are totally ordered as shown in the enumeration, $\mathbf{v_0}$ is the first label. A *database* is a finite set of labelled formulas $\Delta = \{x_1 : A_1, \ldots, x_n : A_n\}$. We assume that

if $x : A \in \Delta$ and $x : B \in \Delta$, then $A = B$.[5]

We use the notation $Lab(E)$ for the set of labels occurring in an expression $E$, and we finally assume that $\mathbf{v_0} \notin Lab(\Delta)$. Label $\mathbf{v_0}$ will be used for queries from the empty database.

DEFINITION 4.2. A query $Q$ is an expression of the form:

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

[4]The names **R**, **E**, **T**, **BCK**, etc. in this chapter refer mainly to *the implicational fragment* of the logical systems known in the literature [Anderson and Belnap, 1975] with the corresponding names. The implicational fragments are usually denoted with the subscript $\rightarrow$. Thus, what we call **R** is denoted in the literature by $\mathbf{R}_\rightarrow$ and so forth; since we are mainly concerned with the implicational systems we have preferred to minimize the notation, stating explicitly when we make exception to this convention.

[5]This restriction will be lifted in Section 3 where conjunction is introduced in the language.

where $\Delta$ is a database, $\delta$ is a finite set of labels not containing $\mathbf{v_0}$; moreover if $x \neq \mathbf{v_0}$ then $x \in Lab(\Delta)$, and $G$ is a formula.

A query from the empty database has the form:

$$\vdash^? \ \mathbf{v_0} : G.$$

Let $\max(\delta)$ denote the maximum label in $\delta$ according to the enumeration of the labels. By convention, we stipulate that if $\delta = \emptyset$, then $\max(\delta) = \mathbf{v_0}$. The set of labels $\delta$ may be thought as denoting the set of resources that are available to prove the goal. Label $x$ in front of the goal has a double role as a 'position' in the database from which the goal is asked, and as available resource.

The rules for success and reduction are parametrized to some conditions $Succ^{\mathbf{S}}$ and $Red^{\mathbf{S}}$ that will be defined below.

- (success)
$$\Delta, \delta \ \vdash^? \ x : q; \text{ succeeds}$$
if $x : q \in \Delta$ and $Succ^{\mathbf{S}}(\delta, x)$.
- (implication) from
$$\Delta, \delta \ \vdash^? \ x : C \to G$$
we step to
$$\Delta \cup \{y : C\}, \delta \cup \{y\} \ \vdash^? \ y : G,$$
where $y > \max(Lab(\Delta))$, (hence $y \notin Lab(\Delta)$);
- (reduction) from
$$\Delta, \delta \ \vdash^? \ x : q,$$
if there is some $z : C \in \Delta$, with $C = A_1 \to \ldots \to A_k \to q$, and there are $\delta_i$, and $x_i$ for $i = 0, \ldots, k$ such that:
  1. $\delta_0 = \{z\}$, $x_0 = z$,
  2. $\bigcup_{i=0}^{k} \delta_i = \delta$,
  3. $Red^{\mathbf{S}}(\delta_0, \ldots, \delta_k, x_0, \ldots, x_k; x)$

  then for $i = 1, \ldots k$, we step to
$$\Delta, \delta_i, \ \vdash^? \ x_i : A_i.$$

The conditions for success are either (s1) or (s2) according to each system:

(s1) $Succ^{\mathbf{S}}(\delta, x) \ \equiv \ x \in \delta$,

(s2) $Succ^{\mathbf{S}}(\delta, x) \ \equiv \ \delta = \{x\}$.

The conditions $Red^{\mathbf{S}}$ are obtained as combination of the following clauses:

(r0) $x_k = x$;

(r1) for $i, j = 0, \ldots, k$, $\delta_i \cap \delta_j = \emptyset$;

(r2) for $i = 1, \ldots, k$, $x_{i-1} \leq x_i$ and $\max(\delta_i) \leq x_i$;

(r3) for $i = 1, \ldots, k$, $x_{i-1} \leq x_i$ and $\max(\delta_i) = x_i$;

(r4) for $i = 1, \ldots, k$, $x_{i-1} < x_i$, $\max(\delta_{i-1}) = x_{i-1} < \min(\delta_i)$ and $\max(\delta_k) = x_k$.

The conditions $Red^{\mathsf{S}}$ are then defined according to Table 2.

Table 4.1. Restrictions on reduction and success.

| Condition | (r0) | (r1) | (r2) | (r3) | (r4) | (Success) |
|---|---|---|---|---|---|---|
| **FL** | $*$ | | | | $*$ | (s2) |
| **T-W** | $*$ | $*$ | | $*$ | | (s2) |
| **T** | $*$ | | | $*$ | | (s2) |
| **E-W** | $*$ | $*$ | $*$ | | | (s2) |
| **E** | $*$ | | $*$ | | | (s2) |
| **L** | | $*$ | | | | (s2) |
| **R** | | | | | | (s2) |
| **BCK** | | $*$ | | | | (s1) |

Notice that

$(r4) \Rightarrow (r3) \Rightarrow (r2)$, and
$(r4) \Rightarrow (r1)$.

We give a quick explanation of the conditions $Succ^{\mathsf{S}}$ and $Red^{\mathsf{S}}$. Remember that $\delta$ are resources which must/can be used.

For the success rule, in all cases but **BCK**, we have that we can succeed if $x : q$ is in the database, $x$ is the only resource left, and $q$ is asked from position $x$; in the case of **BCK** $x$ must be among the available resources, but we do not require that $x$ is the only one left.

The conditions for the reduction rule can be explained intuitively as follows: resources $\delta$ are split in several $\delta_i$, for $i = 1, \ldots, k$ and each part $\delta_i$ must be used in a derivation of a subgoal $A_i$.

In the case of logics *without contraction* we cannot use a resource twice, therefore by restriction (r1), the $\delta_i$s must be disjointed and $z$, the label of the formula we are using in the reduction step, is no longer available.

Restriction (r2) impose that successive subgoals are to be proved from successive positions in the database: only positions $y \geq x$ are 'accessible' from $x$; moreover each $x_i$ must be accessible from resources in $\delta_i$. Notice that the last subgoal $A_k$ must be proved from $x$, the position from which the atomic goal $q$ is asked.

Restriction (r3) is similar to (r4), but it further requires that the position $x_i$ is among the available resources $\delta_i$.

Restriction (r4) forces the goal $A_i$ to be proved by using successive disjointed segments $\delta_i$ of $\delta$. Moreover, $z$ which labels the formula used in the reduction step must be the first (or least) resource among the available ones.

It is not difficult to see that intuitionistic (implicational) logic is obtained by considering success condition (s1) and no other constraint. More interestingly, we can see that **S4**-strict implication is given by considering success condition (s1) and restrictions (r0) and (r2) on reduction. We leave the reader to prove that the above formulation coincides with the database-as-list formulation of **S4** we have seen in the previous chapter. We can therefore consider **S4** as a substructural logic obtained by imposing a restriction on the weakening and the permutation rules. On the other hand, the relation between **S4** and **E** should be apparent: the only difference is the condition on the success rule which controls the weakening restriction.

A few examples of derivations in each system are contained in the proof of the completeness Theorem 4.21.

We introduce some notational conventions we will use later on. We use the notation

$$Q \Rightarrow_z^{Red} Q_1, \ldots, Q_n,$$

to denote that query $Q$ may generate the queries $Q_1, \ldots, Q_n$, by reduction with respect to some

$$z : D_1 \to \ldots \to D_n \to q \in \Gamma.$$

Similarly, let $Q = \Gamma, \gamma \ \vdash^? \ x : A \to B$, if from $Q$ we step to $Q'$ by the implication rule, where

$$Q' = \ \Delta \cup \{y : A\}, \delta \cup \{y\} \ \vdash^? \ y : B,$$

for some $y > \max(Lab(\Delta))$ we write

$$Q \Rightarrow_y^{Imp} Q'.$$

Given a set of labels $\gamma$ and a label $u$, let us define:

$$pred(\gamma, u) = \begin{cases} \max\{z \in \gamma \mid z < u\} \\ \uparrow \ \text{if} \ \neg\exists z \in \gamma \ z < u. \end{cases}$$

$$succ(\gamma, u) = \begin{cases} \min\{z \in \gamma \mid z > u\} \\ \uparrow \ \text{if} \ \neg\exists z \in \gamma \ z > u. \end{cases}$$

By convention, we assume that every inequality in which $pred(\gamma, u)$ ($succ(\gamma, u)$) is one of the two terms trivially holds if $pred(\gamma, u) =\uparrow$ ($succ(\gamma, u) =\uparrow$); similarly, we assume that an inequality such as $\max\{pred(\gamma, u), x\} < y$ reduce to $x < y$, whenever $pred(\gamma, u)$ is undefined.

We list some properties of the proof system which allow some simplification. The proofs are by induction on the height of successful derivations and are mostly omitted. Unless stated otherwise the propositions hold for any system.

PROPOSITION 4.3. *If* $\Gamma, \gamma \ \vdash^? \ x : B$ *succeeds,* $u : A \in \Gamma$*, but* $u \notin \gamma$*, then*

$$\Gamma - \{u : A\}, \gamma \ \vdash^? \ x : B \text{ succeeds.}$$

By this proposition it is apparent that $\gamma$ specifies usage. Resources which are not included in $\gamma$ are not relevant for the computation and the formulas they label can be omitted.

PROPOSITION 4.4. *If* $\Gamma, \gamma \ \vdash^? \ x : B$ *succeeds, then* $\gamma \subseteq Lab(\Gamma)$.

By the previous proposition, when we consider successful queries we can assume that $\gamma \subseteq Lab(\Gamma)$, and hence $\max(\gamma) \leq \max(Lab(\Gamma))$.

The following proposition generalizes the success rule to the identity property.

PROPOSITION 4.5. *If* $x : A \in \Gamma$*, and* $Succ^{\mathbf{S}}(\gamma, x)$*, then* $\Gamma, \gamma \ \vdash^? \ x : A$ *succeeds.*

PROPOSITION 4.6. *If*

1. $\Gamma, \gamma \ \vdash^? \ x : B$ *succeeds,* $\Gamma \subseteq \Delta$*, then*
2. $\Delta, \gamma \ \vdash^? \ x : B$ *succeeds.*

*In the case of* **BCK** *if* $\gamma \subseteq \delta$ *also*

3. $\Delta, \delta \ \vdash^? \ x : B$ *succeeds.*

*Moreover, for any successful derivation of 1. there is a corresponding derivation of 2. (viz. 3.) of no greater height.*

PROPOSITION 4.7.

- *For* **R**, **L**, **BCK**, *if* $\Gamma, \gamma \ \vdash^? \ x : B$ *succeeds, then*

  *for every* $y \in \gamma$, $\Gamma, \gamma \ \vdash^? \ y : B$ *succeeds.*

- *For* **R**, **L**, **BCK**, **E**, **E-W**, *if* $\Gamma, \emptyset \ \vdash^? \ x : B$ *succeeds, then*

  *for every* $y$, $\Gamma, \emptyset \ \vdash^? \ y : B$ *succeeds.*

- *For* **E**, **E-W**, *if* $\Gamma, \gamma \ \vdash^? \ x : B$ *succeeds and* $x \notin \gamma$*, then,*

  *for every* $y \geq \max(\gamma)$, $\Gamma, \gamma \ \vdash^? \ y : B$ *succeeds.*

*Moreover, the height of a successful derivation does not increase.*

Because of the previous proposition, in the case of **R**, **L**, and **BCK** we can omit the label $x$ in front of the goal. However, we do not make use of this and other possible simplifications at this stage since we want to carry on a uniform development.

DEFINITION 4.8. We say that a query

$$\Gamma, \gamma \vdash^? x : A$$

is **S**-*regular*, where **S** in one of the logic under consideration if the following holds:

- for **R**, **L**, **BCK**, if $\gamma \neq \emptyset$, then $x \in \gamma$;
- for **E**, **E-W**, $\max(\gamma) \leq x$;
- for **T**,**T-W**, **FL**, $\max(\gamma) = x$ (thus if $\gamma = \emptyset$ then $x = \mathbf{v_0}$).

An **S**-*regular* derivation is a derivation in which every query is **S**-regular.

Notice that a query from the empty database is **S**-regular for every system **S**. Without loss of generality, we can restrict our attention to regular queries and regular derivations.

PROPOSITION 4.9. *Let $Q$ be an* **S**-*regular query, if $Q$ succeeds, then it succeeds by an* **S**-*regular derivation.*

From now on we restrict our consideration to regular queries and regular derivations. The main reason is technical: this restriction helps in the proof of cut-admissibility contained in the next section.

We conclude this section by showing that successful queries are closed under *formula substitution*. The property of closure under substitution is not to be taken for granted, as the deduction rules pay attention to the form of the formulas, that is whether they are atomic or not.

THEOREM 4.10. *If $Q = \Gamma, \gamma \vdash^? x : A$ succeeds, then also $Q' = \Gamma[q/B], \gamma \vdash^? x : A[q/B]$ succeeds.*

## *2.1 Admissibility of Cut*

In this section we prove the admissibility of cut for all the logics we have considered so far. We first have to deal with the notion of substitution of formulas and labels by databases. In this case the proof of the admissibility of cut is more complex than in the cases we have seen in the previous chapters. The reason is that the deduction rules are more constrained than in the previous cases. In particular we have to take into account the ordering of formulas in the databases involved in a cut inference step.

Since we treat the implicational fragment in this section, we need the notion of compatibility with respect to substitution that we have already met in Chapter 4.

DEFINITION 4.11. Given two databases $\Gamma$, with $u : A \in \Gamma$ and $\Delta$, we say that $\Gamma$ and $\Delta$ are *compatible for substitution* if $z \in Lab(\Gamma) \cap Lab(\Delta)$, then $z : C \in \Gamma$ iff $z : C \in \Delta$, and whenever they are compatible for substitution, we let:

$$\Gamma[u : A/\Delta] = (\Gamma - \{u : A\}) \cup \Delta.$$

Given two sets of resources $\gamma$ and $\delta$ and a label $u$ we define:

$$\gamma[u/\delta] = \begin{cases} (\gamma - \{u\}) \cup \delta \text{ if } u \in \gamma \\ \gamma \text{ otherwise.} \end{cases}$$

By cut we mean the following inference rule: If

$$\Gamma[u : A], \gamma \ \vdash^{?} \ x : B \quad \text{and} \quad \Delta, \delta \ \vdash^{?} \ y : A$$

succeed, then also

$$\Gamma[u : A/\Delta], \gamma[u/\delta] \ \vdash^{?} \ x[u/y] : B \text{ succeeds.}$$

However, unless we put some constraint on $\gamma$ and $\delta$, the above rule does not hold. To give a simple example, let us consider the case of $\mathbf{L}$ (linear implication), we have:

1. $\{x_2 : p \to q, \mathbf{x_3} : \mathbf{p}\},\ \{x_2, x_3\} \ \vdash^{?} \ x_3 : q$ and
2. $\{x_1 : (p \to q) \to p, x_2 : p \to q\},\ \{x_1, x_2\} \ \vdash^{?} \ \mathbf{x_2} : \mathbf{p}$ both succeed, but
3. $\{x_1 : (p \to q) \to p, x_2 : p \to q\},\ \{x_1, x_2\} \ \vdash^{?} \ x_2 : q$ fails.

Here we want to replace $\mathbf{x_3} : \mathbf{p}$ occurring in 1. by the database in 2. . Since the database shares resource $x_2$, $x_2 : p \to q$ must be used *twice* in a deduction of 3., and this is not allowed by the condition for $\mathbf{L}$-computation. In this case, the restriction we must impose is that the two resource sets (in the two premises of the cut) $\gamma$ and $\delta$ are *disjoint*. For other logics, for instance, where the order matters, we must ensure that the substitution of $\delta$ in $\gamma$ respects the ordering of the labels in $\gamma$. However, we can always fulfill these compatibility constraints by renaming the labels in the proper way. In the previous example, for instance, we can rename the labels in the first query, and get

1′. $\{\mathbf{y_2} : p \to q, \mathbf{y_3} : \mathbf{p}\},\ \{y_2, y_3\} \ \vdash^{?} \ y_3 : q$

by cut we now obtain:

3′. $\{x_1 : (p \to q) \to p, x_2 : p \to q, \mathbf{y_2} : p \to q\}\ \{x_1, x_2, y_2\} \ \vdash^{?} \ x_2 : q$

that succeeds.

Hence we have two alternatives: either we accept the above formulation of the cut rule, but we restrict its applicability to queries which satisfy specific conditions for each system; or we find a more general notion of cut that holds in every case.

As we will see, we can define the more general notion of cut by incorporating the needed re-labelling. However, we will prove the admissibility of the cut in the restricted form since, in some sense it is more informative. For instance, from the above premises 1. and 2., we can legitamately infer the conclusion 3. in **R**, **T**, **E**, and not only the weaker $3'$.

We introduce some notation to simplify the following development. Given two queries $Q$, $Q'$, with $Q = \Gamma[u : A], \gamma \vdash^? x : A$ and $Q' = \Delta, \delta \vdash^? y : A$, no matter whether they are compatible or not, the result of cutting $Q$ by $Q'$ on $u : A$ is uniquely determined, and we denote it by the query

$$Q'' = CUT(Q, Q', u).$$

DEFINITION 4.12 (Compatibility for Cut). Given two queries

$$Q = \Gamma[u : A], \gamma \vdash^? x : A \text{ and } Q' = \Delta, \delta \vdash^? y : A,$$

we say that $Q$ and $Q'$ are *compatible for cut* on position/resource $u$ in system **S**, denoting this by $COMP^{\mathbf{S}}(Q, Q', u)$,[6] if the following holds:

- $\Gamma$ and $\Delta$ are compatible for substitution;
- the following combinations of the conditions (c1)– (c3) below hold.

  (c1) $\gamma \cap \delta = \emptyset$;

  (c2) $pred(\gamma, u) < \min(\delta)$ and $\max(\delta) < succ(\gamma, u)$;

  (c3) $pred(\gamma, u) \leq y$, and if $u < x$, then $y \leq \min\{succ(\gamma, u), x\}$.

  - for **R**: nothing;
  - for **L**, **BCK**: (c1);
  - for **FL**: (c2);
  - for **T**, **E**: (c3);
  - for **T-W**, **E-W**: (c1) and (c3).

Here is an explanation of the above conditions: condition (c1) says that the resources of the two premises are disjoint; this condition must be assumed for systems without contraction. Condition (c2) is stronger and requires that $\delta$, seen as an ordered sequence of labels, must fit in between the predecessor and the successor of the label $u$ that we replace by cut; in other words, $u$ can be replaced by the entire 'segment' $\delta$ without mixing $\gamma$ and $\delta$. Condition (c3) is a weakening of (c2) allowing merging of $\gamma$ and $\delta$ up to the successor of the cut formula label $u$. Condition (c3) says exactly this when $succ(\gamma, u)$ exists (whence $succ(\gamma, u) \leq x$)

[6]As can be seen from the conditions below, compatibility is a relation on (sets of) labels which occur in a query, the formulas do not play any role. In Section 3 we wil use the alternative notation $COMP^{\mathbf{S}}(\Gamma, \gamma, x, u; \Delta, \delta, y)$.

and $y = \max(\delta)$. It is stated in more general terms, since $succ(\gamma, u)$ may not exist, and in the cases of **E** and **E-W** it may be $y \geq \max(\delta)$. For instance, let $x_1 < x_2 < \ldots < x_9$ and let us consider two queries with $\gamma = \{x_2, x_4, x_6, x_8, x_9\}$, $u = x_6$, $x = x_9$, and $\delta = \{x_1, x_3, x_6, x_7\}$, $y = x_7$, satisfy condition (c3). On the other hand, if $\delta = \{x_1, x_3, x_6, x_9\}$ and $y = x_9$, the two queries do not satisfy condition (c3).

It should also be clear that if $y = \max(\delta)$ then condition (c2) implies both condition (c1) and condition (c3).

The compatibility conditions preserve the regularity of the queries.

PROPOSITION 4.13. *If $Q$ and $Q'$ are* **S***-regular and $COMP^{\mathbf{S}}(Q, Q', u)$ holds, then the query $Q'' = CUT(Q, Q', u)$ is* **S***-regular.*

The proof of the admissibility of cut relies on the lemmas below, which state properties and relations of $COMP^{\mathbf{S}}$, $Red^{\mathbf{S}}$, $Succ^{\mathbf{S}}$.

We give an intuitive explanation of the lemmas. Lemma 4.14 deals with the case when the cut formula is an atom and it is used for immediate success. Lemmas 4.15 and 4.16 ensure the permutability of cut with the implication rule and with the reduction rule, respectively. Lemma 4.17 deals with the case when the cut formula is used in a reduction step, that is, it is its principal formula. To explain the role of Lemmas 4.16 and 4.17, we illustrate the cut elimination process in some detail in this last case; we do not mention the labels, but we focus on the inference steps. Let us have

1. $\Gamma, D \to q \ \vdash^{?} \ q$ and  2. $\Delta \ \vdash^{?} \ D \to q$.

From 1. we step to

3. $\Gamma, D \to q \ \vdash^{?} \ D$.

By cutting 2. and 3. we obtain

4. $\Gamma, \Delta \ \vdash^{?} \ D$ succeeds.

By 2. we get:

5. $\Delta, D \ \vdash^{?} \ q$ succeeds.

Now we cut 5. and 4. on $D$ and obtain:

6. $\Gamma, \Delta \ \vdash^{?} \ q$ succeeds.

Since, we have labels and constraints, we cannot freely perform cut, we must ensure that the compatibility constraints are satisfied. The meaning of the lemmas is that: by 4.16 if 1. and 2. are compatible for cut, then so are 3. and 2.; by 4.17 if 1. and 2. are compatible for cut, then so are 5. and 4. . The structure of the cut elimination process is fixed and it is exactly the same as in the previous chapters. Much of the effort is to ensure the right propagation of compatibility constraints. We assume that the queries are **S**-regular.

LEMMA 4.14. *If* $\Delta, \delta \vdash^? y : q$ *succeeds and* $Succ^{\mathsf{S}}(\gamma, u)$ *holds, then* $\Delta$, $\gamma[u/\delta] \vdash^? y : q$ *succeeds.*

LEMMA 4.15. *If* $Q \Rightarrow_z^{Imp} Q_1$, *and* $COMP^{\mathsf{S}}(Q, Q', u)$, *then there exists a label* $z'$, *such that* $Q \Rightarrow_{z'}^{Imp} Q_1[z/z']$, $COMP^{\mathsf{S}}(Q_1[z/z'], Q', u)$, *and*

$$CUT(Q, Q', u) \Rightarrow_{z'}^{Imp} CUT(Q_1[z/z'], Q', u).$$

LEMMA 4.16. *Suppose* $Q \Rightarrow_z^{Red} Q_1, \ldots, Q_n$, *where*

$Q = \Gamma[u : A], \gamma \vdash^? x : q$
$Q_i = \Gamma[u : A], \gamma_i \vdash^? x_i : D_i$, *and* $Q' = \Delta, \delta \vdash^? y : A$.

*If* $COMP^{\mathsf{S}}(Q, Q', u)$ *then*

- *(a)* $COMP^{\mathsf{S}}(Q_i, Q', u)$;
- *(b)* $z \neq u$ *implies* $CUT(Q, Q', u) \Rightarrow_z^{Red} CUT(Q_1, Q', u), \ldots, CUT(Q_n, Q', u)$.

LEMMA 4.17. *Let the following hold:*

*(i)* $Q = \Gamma[u : D_1 \to \ldots \to D_n \to q], \gamma \vdash^? x : q$,

*(ii)* $Q' = \Delta, \delta \vdash^? y : D_1 \to \ldots \to D_n \to q$,

*(iii)* $Q \Rightarrow_u^{Red} Q_1, \ldots, Q_n$,

*(iv)* $COMP^{\mathsf{S}}(Q, Q', u)$,

*(v)* $Q'_i = CUT(Q_i, Q', u)$ *for* $i = 1, \ldots, n$,

*(vi)* $Q''_0 = \Delta \cup \{z_1 : D_1, \ldots, z_n : D_n\}, \delta \cup \{z_1, \ldots, z_n\} \vdash^? z_n : q$, *where*

$$\max\{\max(\gamma), \max(\delta), x, y, u\} < z_1 < z_2 < \ldots < z_n,$$

*(vii)* $Q''_i = CUT(Q''_{i-1}, Q'_i, z_i)$ *for* $i = 1, \ldots, n$,

*then,* $COMP^{\mathsf{S}}(Q''_{i-1}.Q'_i, z_i)$ *holds for* $i = 1, \ldots, n$.

THEOREM 4.18 (Admissibility of Cut). *Let* $Q = \Gamma[u : A], \gamma \vdash^? x : B$ *and* $Q' = \Delta, \delta \vdash^? y : A$ *be regular queries. If* $Q$ *and* $Q'$ *succeed and* $COMP^{\mathsf{S}}(Q, Q', u)$, *then also*

$Q^* = CUT(Q, Q', u) = \Gamma[u : A/\Delta], \gamma[u/\delta] \vdash^? x[u/y] : B$ *succeeds.*

**Proof.** Let $Q$ and $Q'$ be as in the statement of the theorem. As usual, we proceed by double induction on pairs $(c, h)$, where $c$ is the complexity of the cut formula $A$ and $h$ is a successful derivation of $Q$. We give the details for the case of immediate success ($c = h = 0$) and for the case of reduction with respect to $u : A$ ($c, h > 0$). The other cases follow by the induction hypothesis, using Lemma 4.15 if the first step is by the implication rule, and Lemma 4.16 if the first step is by the reduction rule.

Let $c = h = 0$, then $Q$ immediately succeeds, that is $B$ is an atom $q$, $x : q \in \Gamma$ and $Succ^{\mathsf{S}}(\gamma, x)$ holds. If $x \neq u$, then $x : q$ is also in $\Gamma[u : A/\Delta]$, moreover $x[u/y] = x$ and $\gamma[u/\delta] = \gamma$. By Propositions 4.3 and 4.6 we have that $Q^* = \Gamma - \{u : A\} \cup \Delta, \gamma[u/\delta] \vdash^? x[u/y] : B$ succeeds. If $x = u$, then $x[u/y] = y$; since $Q'$ succeeds, by Lemma 4.14 we get that $Q^*$ succeeds.

Let $c, h > 0$ and $Q$ succeed by reduction with respect to $u : A$. Let $A = D_1 \to \ldots \to D_n \to q$. We have $Q \Rightarrow_u^{Red} Q_1, \ldots, Q_n$, and each $Q_i$ succeeds with smaller height; by Lemma 4.16 we have $COMP^{\mathsf{S}}(Q_i, Q', u)$, hence by the induction hypothesis we get that $Q'_i = CUT(Q_i, Q', u)$ succeeds. Now let,

$$Q_0^* = \Delta \cup \{z_1 : D_1, \ldots, z_n : D_n\}, \delta \cup \{z_1, \ldots, z_n\} \vdash^? z_n : q,$$
where $\max\{\max(\gamma), \max(\delta), x, y, u\} < z_1 < z_2 < \ldots z_n$;

by Lemma 4.17 we have $COMP^{\mathsf{S}}(Q_0^*, Q'_1, z_1)$, notice that $cp(D_1) < cp(A) = c$, thus we may apply the induction hypothesis and obtain that $Q_1^* = CUT(Q_0^*, Q'_1, z_1)$ that is

$$\begin{aligned} Q_1^* \quad = \quad & \Gamma - \{u : A\} \cup \Delta \cup \{z_2 : D_2, \ldots, z_n : D_n\}, \\ & \gamma_1[u/\delta] \cup \delta \cup \{z_2, \ldots, z_n\} \vdash^? z_n : q \end{aligned}$$

succeeds. By Lemma 4.17 also $COMP^{\mathsf{S}}(Q_1^*, Q'_2, z_2)$, and since $cp(D_2) < cp(A)$, we can cut $Q_1^*$ and $Q'_2$ on $z_2 : D_2$ and obtain

$$\begin{aligned} Q_2^* \quad = \quad & \Gamma - \{u : A\} \cup \Delta \cup \{z_3 : D_3, \ldots, z_n : D_n\}, \\ & \gamma_1[u/\delta] \cup \gamma_2[u/\delta] \cup \delta \cup \{z_3, \ldots, z_n\} \vdash^? z_n : q \end{aligned}$$

succeeds. by repeating this argument up to $n$ (using Lemma 4.17), we finally get

$$Q_n^* = \Gamma - \{u : A\} \cup \Delta, \gamma_1[u/\delta] \cup \ldots \cup \gamma_n[u/\delta] \cup \delta \vdash^? z_n[z_n/x_n[u/y]] : q$$

succeeds. Since $x_n = x$ and $u \in \gamma$, we have

$$\gamma[u/\delta] = (\bigcup_{i=0}^{n} \gamma_i)[u/\delta] = \bigcup_{i=0}^{n} \gamma_i[u/\delta] \cup \delta.$$

Hence $Q_n^*$ is $Q^*$ and we are done. ■

COROLLARY 4.19 (Modus Ponens). *If*

$\vdash^? \mathbf{v_0} : A \to B$ *and* $\vdash^? \mathbf{v_0} : A$ *succeed,* *then also* $\vdash^? \ \mathbf{v_0} : B$ *succeeds.*

**Proof.** If $\vdash^? \ \mathbf{v_0} : A \to B$ succeed, then also $Q = \ u : A, \{u\} \ \vdash^? \ u : B$ (with $u > \mathbf{v_0}$) succeeds by the implication rule. It easily seen that $Q$ and $Q' = \ \ \vdash^? \ \mathbf{v_0} : A$ are compatible for cut in every system. Thus, by cut we get that $\vdash^? \ \mathbf{v_0} : B$ succeeds. ■

*A general form of Cut*

We have proved the admissibility of cut under specific compatibility conditions for each system. It is easily seen that we can always rename the labels in a query in order to match the compatibility constraints.

But we can do more: we can incorporate the necessary relabellings in the definition of cut in order to achieve a formulation which holds for every system without restrictions. To this end, given $Q_1 = \Gamma[u : A], \gamma \ \vdash^? \ x : B$ and $Q_2 = \Delta, \delta \ \vdash^? \ y : A$, let us call a pair $(\psi_1, \psi_2)$ of label substitutions *suitable for cutting* $Q_1$ *by* $Q_2$ *on* $u$ if:

- $\psi_1$ and $\psi_2$ are order-preserving on $\gamma \cup \{u, x\}$ and on $\delta \cup \{y\}$, respectively;[7]
- $pred(\gamma\psi_1, \psi_1(u)) < \min(\delta\psi_2)$;
- $\psi_2(y) < \min\{succ(\gamma\psi_1, \psi_1(u)), \psi_1(x)\}$.

Now the cut rule becomes

> if $Q_1 = \Gamma[u : A], \gamma \ \vdash^? \ x : B$ and $Q_2 = \Delta, \delta \ \vdash^? \ y : A$ succeed

then also

> $\Gamma\psi_1[\psi_1(u) : A/\Delta\psi_2], \gamma[\psi_1(u)/\delta\psi_2] \ \vdash^? \ \psi_1(x)[\psi_1(u)/\psi_2(y)] : B$
> suceeds,

where $(\psi_1, \psi_2)$ is any pair of label substitution *suitable for cutting* $Q_1$ *by* $Q_2$ *on* $u$. Given any query $Q = \Gamma, \gamma \ \vdash^? \ x : A$ and a label substitution $\psi$, $Q\psi$ is $\Gamma\psi, \gamma\psi \ \vdash^? \ \psi(x) : A$, thus the formula in the conclusion is nothing other than that obtained by cutting (in the sense of Theorem 4.18) $Q_1\psi_1$ by $Q_2\psi_2$ on $\psi_1(u)$. It is easy to see that this rule is admissible by Theorem 4.18. We simply observe that if $(\psi_1, \psi_2)$ are *suitable for cutting* $Q_1$ *by* $Q_2$ *on* $u$ then

- $Q_i$ succeeds implies $Q_i\psi_i$ succeeds for $i = 1, 2$;
- $COMP^{\mathbf{S}}(Q_1\psi_1, Q_2\psi_2, \psi_1(u))$ holds for any system **S**.

Then the result follows by Theorem 4.18.

[7]We say that a label substitution $\psi$ is order-preserving on a set $I$ if $z_1, z_2 \in I$ and $z_1 < z_2$ implies $\psi(z_1) < \psi(z_2)$.

## 2.2 Soundness and Completeness

We prove first the completeness of the proof procedure with respect to the Hilbert axiomatization of each system of Section 2.

DEFINITION 4.20 (Axiomatization of implication). We consider the following list of axioms:

(id) $A \to A$;

(h1) $(B \to C) \to (A \to B) \to A \to C$;

(h2) $(A \to B) \to (B \to C) \to A \to C$;

(h3) $(A \to A \to B) \to A \to B$;

(h4) $(A \to B) \to ((A \to B) \to C) \to C$;

(h5) $A \to (A \to B) \to B$;

(h6) $A \to B \to B$.

Together with the following rules:

$$\frac{A \to B \qquad A}{B}\,(MP)$$

$$\frac{A \to B}{(B \to C) \to A \to C}\,(\mathit{Suff}).$$

Each system is axiomatized by taking the closure under modus ponens (MP) and under substitution of the combinations of axioms/rules of Table 5.2.

Table 4.2. Axioms for substructural implication.

| Logic | Axioms |
|---|---|
| **FL** | (id), (h1), (*Suff*) |
| **T-W** | (id), (h1), (h2) |
| **T** | (id), (h1), (h2), (h3) |
| **E-W** | (id), (h1), (h2), (h4) |
| **E** | (id), (h1), (h2), (h3), (h4) |
| **L** | (id), (h1), (h2), (h5) |
| **R** | (id), (h1), (h2), (h3), (h5) |
| **BCK** | (id), (h1), (h2), (h5), (h6) |
| **I** | (id), (h1), (h2), (h3), (h5), (h6) |

In the above axiomatization, we have not worried about the minimality and independence of the group of axioms for each system. For some systems the corresponding list of axioms given above is redundant, but it quickly shows some inclusion relations among the systems. We just remark that in presence of (h4), (h2) can be obtained by (h1). Moreover, (h4) is a weakening of (h5). The rule of (*Suff*) is clearly obtainable from (h2) and (MP). To have a complete picture we have included also intutionistic logic **I**, although the axiomatization above is highly redundant (see Chapter 2, Section 1).

We prove that each system is complete with respect to the respective axiomatization. By Corollary 4.19 and Theorem 4.10, all we have to prove is that any atomic instance of each axiom succeeds in the relative proof system. In addition, we need to show the admissibility of (*Suff*) rule for **FL**.

THEOREM 4.21 (Completeness). *For every system* **S**, *if $A$ is a theorem of* **S**, *then* $\vdash \mathbf{v_0} : A$ *succeeds in the corresponding proof system for* **S**.

**Proof.** We show a derivation of an arbitrary atomic instance of each axiom in the relative proof system. In the case of reduction, the condition $\gamma = \bigcup \gamma_i$, will not be explicitly shown, as its truth will be apparent by the choice of $\gamma_i$. We assume that the truth of the condition for the success rule is evident and we do not mention it. At each step we only show the current goal, the available resources and the *new* data introduced in the database, if any. Moreover, we justify the queries obtained by a reduction step by writing the relation $Red^{\mathbf{S}}(\gamma_0, \ldots, \gamma_n, x_0, \ldots, x_n; x)$ (for suitable $\gamma_i,\ x_i$) under them; the database formula used in the reduction step is identified by $\gamma_0$.

(id) In all systems:

$$\vdash^? \ \mathbf{v_0} : a \rightarrow a$$

we step to

$$u : a, \{u\} \ \vdash^? \ u : a,$$

which immediately succeeds in all systems.

(h1) In all systems:

$$\vdash^? \ \mathbf{v_0} : (b \rightarrow c) \rightarrow (a \rightarrow b) \rightarrow a \rightarrow c.$$

three steps of the implication rule leads to:

$$\begin{array}{rcl}
x_1 : b \rightarrow c,\ x_2 : a \rightarrow b,\ x_3 : a,\ \{x_1, x_2, x_3\} & \vdash^? & x_3 : c \\
\{x_2, x_3\} & \vdash^? & x_3 : b \\
Red^{\mathbf{S}}(\{x_1\}, \{x_2, x_3\}, x_1, x_3; x_3) & & \\
\{x_3\} & \vdash^? & x_3 : a \\
Red^{\mathbf{S}}(\{x_2\}, \{x_3\}, x_2, x_3; x_3). & &
\end{array}$$

(h2) In all systems, but **FL**:

$$\vdash^? \ \mathbf{v_0} : (a \rightarrow b) \rightarrow (b \rightarrow c) \rightarrow a \rightarrow c.$$

three steps of the implication rule leads to:

$$\begin{array}{rcl}
x_1 : a \to b,\ x_2 : b \to c,\ x_3 : a,\ \{x_1, x_2, x_3\} & \vdash^? & x_3 : c \\
(*)\ \{x_1, x_3\} & \vdash^? & x_3 : b, \\
Red^{\mathbf{S}}(\{x_2\}, \{x_1, x_3\}, x_2, x_3; x_3) & & \\
\{x_3\} & \vdash^? & x_3 : a, \\
Red^{\mathbf{S}}(\{x_1\}, \{x_3\}, x_1, x_3; x_3). & &
\end{array}$$

the step (*) is allowed in all systems, but those with (r4), namely **FL**.

(h3) In all systems, but those with (r1) or (r4):

$$\vdash^? \ \mathbf{v_0} : (a \to a \to b) \to a \to b.$$

Two steps of the implication rule leads to:

$$x_1 : a \to a \to b,\ x_2 : a,\ \{x_1, x_2\} \vdash^? \ x_2 : b.$$

By reduction we step to:

$$\{x_2\} \vdash^? \ x_2 : a \quad \text{and} \quad \{x_2\} \vdash^? \ x_2 : a$$

since $Red^{\mathbf{S}}(\{x_1\}, \{x_2\}, \{x_2\}, x_1, x_2, x_2; x_2)$ holds in all systems without (r1) and (r4).

(h4) In all systems, but those with (r3) or (r4):

$$\vdash^? \ \mathbf{v_0} : (a \to b) \to ((a \to b) \to c) \to c.$$

two steps of the implication rule leads to:

$$\begin{array}{rcl}
x_1 : a \to b,\ x_2 : (a \to b) \to c,\ \{x_1, x_2\} & \vdash^? & x_2 : c \\
(*)\ \{x_1\} & \vdash^? & x_2 : a \to b \\
Red^{\mathbf{S}}(\{x_2\}, \{x_1\}, x_2, x_2; x_2) & & \\
x_3 : a,\ \{x_1, x_3\} & \vdash^? & x_3 : b \\
\{x_3\} & \vdash^? & x_3 : a \\
Red^{\mathbf{S}}(\{x_1\}, \{x_3\}, x_1, x_3; x_3) & &
\end{array}$$

The step (*) is allowed by (r2), but not by (r3) or (r4) since $\max(\{x_1\}) = x_1 < x_2$.

(h5) In **L**,**R**,**BCK**:

$$\vdash^? \ \mathbf{v_0} : a \to (a \to b) \to b.$$

two steps of implication rule leads to:

$$\begin{array}{rcl}
x_1 : a,\ x_2 : a \to b,\ \{x_1, x_2\} & \vdash^? & x_2 : b \\
\{x_1\} & \vdash^? & x_1 : a \\
Red^{\mathbf{S}}(\{x_2\}, \{x_1\}, x_2, x_1; x_2). & &
\end{array}$$

(h6) In **BCK** we have:

$$\vdash^? \mathbf{v_0} : a \to b \to b.$$

two steps by implication rule leads to:

$$x_1 : a,\ x_2 : b,\ \{x_1, x_2\} \vdash^? x_2 : b$$

which succeeds by the success condition of **BCK**. This formula does not succeed in any other system.

(*Suff*) We prove the admissibility of (*Suff*) rule in **FL**. Let $\vdash^? \mathbf{v_0} : A \to B$ succeed. Then for any formula $C$, we have to show that

$\vdash^? \mathbf{v_0} : (B \to C) \to A \to C$ succeeds.

Let $C = C_1 \to \ldots \to C_n \to q$. Starting from

$$\vdash^? \mathbf{v_0} : (B \to C_1 \to \ldots \to C_n \to q) \to A \to C_1 \to \ldots \to C_n \to q$$

by the implication rule, we step to

$$\Delta, \{x_1, \ldots, x_{n+2}\} \vdash x_{n+2} : q,$$

where

$$\begin{aligned}\Delta \ = \ & \{x_1 : B \to C_1 \to \ldots \to C_n \to q,\ x_2 : A, \\ & x_3 : C_1, \ldots, x_{n+2} : C_n\}.\end{aligned}$$

From the above query we step by reduction to:

$Q' = \Delta, \{x_2\} \vdash^? x_2 : B$ and
$Q_i = \Delta, \{x_{i+2}\} \vdash^? x_{i+2} : C_i$ for $i = 1, \ldots, n$.

since the condition for reduction are satisfied. By hypothesis, $\vdash^? \mathbf{v_0} : A \to B$ succeeds, which implies, by the implicational rule, that $x_2 : A, \{x_2\} \vdash^? x_2 : B$ succeeds, but then $Q'$ succeeds by monotony. Queries $Q_i$ succeed by Proposition 4.5. ■

*Soundness with Respect to Fine Semantics*

We prove the soundness of the proof procedure with respect to a possible world semantics for substructural implicational logics. The first possible world semantics for substructural logics has been elaborated by Urquhart [1972]; according to [Anderson and Belnap, 1975] similar ideas were developed by Routley and by Fine at about the same time. The intuitive idea of Urquhart's semantics is to consider possible worlds equipped by a composition operation, that is forming an algebraic structure. We may think of the elements of the algebraic structure as pieces of information or resources which can be combined along a deduction. In Urquhart semantics the algebraic structure is a semilattice. The semantics by Urquhart can be interestingly seen as a sort of metalevel codification of the bookkeeping mechanism used to control natural deduction proofs in relevant logics [Anderson and Belnap, 1975]. This semantical construction works well for the implicational fragment of many systems, first of all **R**, but it cannot cover *uniformly* all the systems we are considering in this chapter. For instance in the case of **E** we need to consider two-dimensional models.

Another problem of Urquhart's semantics is that it does not work for the language containing disjunction or negation, we will come back to this point in the next section. The semantics we adopt in this section is a simplification of the one proposed in [Fine, 1974], [Anderson *et al.*, 1992] and elaborated more recently by Došen [1988; 1989]. [8]

Routley and Meyer [Routley *et al.*, 1982] have developed a possible world semantics for substructural logics based on a three-place accessibility relation. Their semantics works uniformly for all systems and the whole propositional language. In the next section we will introduce it to prove the completeness of our procedures for a broader fragment. Although semantics is not our main concern, we observe that all these semantic constructions are related; we refer to [Anderson *et al.*, 1992], paragraph 51.5 in particular, for more details (see also [Došen, 1989] and [Ono, 1993] where a more general semantics based on residuated lattices is developed).

DEFINITION 4.22. Let us fix a language $\mathcal{L}$, a Fine **S**-structure[9] $M$ is a tuple of the form:

$$M = (W, \leq, \circ, 0, V),$$

where $W$ is a non empty set, $\circ$ is a binary operation on $W$, $0 \in W$, $\leq$ is a partial order relation on $W$, $V$ is a function of type $W \rightarrow Pow(Var)$. In all structures the following properties are assumed to hold:

$0 \circ a = a$,
$a \leq b$ implies $a \circ c \leq b \circ c$,
$a \leq b$ implies $V(a) \subseteq V(b)$.

For each systems **S**, a **S**-structure satisfies a subset of the following conditions, as specified in Figure 4.22

(a1) $a \circ (b \circ c) \leq (a \circ b) \circ c$;

(a2) $a \circ (b \circ c) \leq (b \circ a) \circ c$;

(a3) $(a \circ b) \circ b \leq a \circ b$;

(a4) $a \circ 0 \leq a$;

[8] Dealing only with the implicational fragment, we have simplified Fine semantics: we do not have *prime* or maximal elements.

[9] We just write **S**-structure if there is no risk of confusion.

(a5) $a \circ b \leq b \circ a$;

(a6) $0 \leq a$.

**Truth conditions** for $a \in W$, we define

- $M, a \models p$ if $p \in V(a)$;
- $M, a \models A \to B$ if

$$\forall b \in W(M, b \models A \ \Rightarrow \ M, a \circ b \models B).$$

We say that $A$ is *valid* in $M$ (denoted by $M \models A$) if $M, 0 \models A$. We say that $A$ is **S**- valid, denoted by $\models_{\mathbf{S}}^{Fine} A$ if $A$ is valid in every **S**-structure.

Table 4.3. Algebraic conditions of Fine Semantics.

| Logic | (a1) | (a2) | (a3) | (a4) | (a5) | (a6) |
|---|---|---|---|---|---|---|
| **FL** | $*$ | | | | | |
| **T-W** | $*$ | $*$ | | | | |
| **T** | $*$ | $*$ | $*$ | | | |
| **E-W** | $*$ | $*$ | | $*$ | | |
| **E** | $*$ | $*$ | $*$ | $*$ | | |
| **L** | $*$ | $*$ | | $*$ | $*$ | |
| **R** | $*$ | $*$ | $*$ | $*$ | $*$ | |
| **BCK** | $*$ | $*$ | | $*$ | $*$ | $*$ |
| **I** | $*$ | $*$ | $*$ | $*$ | $*$ | $*$ |

We have included the intutionistic logic **I** in Figure 5.3 to show its proper place within this framework. Again this list of semantical conditions is deliberately redundant in order to show quickly the inclusion relation among the systems. The axiomatization given at the beginning of the section is sound and complete with respect to this semantics. In particular each axiom (h**i**) corresponds to the semantical condition (a**i**).

THEOREM 4.23 (Anderson *et al.*, 1992, Fine, 1974, Došen, 1989). $\models_{\mathbf{S}} A$ *if and only if $A$ is derivable in the corresponding axiom system of Definition 4.20.*

The following lemma, (whose proof is left to the reader), shows that the hereditary property extends to all formulas.

LEMMA 4.24. *Let $M = (W, \leq, \circ, 0, V)$ be an* **S**-*structure, let $a, b \in W$, then for any formula $A$:*

*if $M, a \models A$ and $a \leq b$, then $M, b \models A$.*

We assume that $\circ$ associates to the left, so we write

$$a \circ b \circ c = (a \circ b) \circ c.$$

In order to prove the soundness of our procedure, we need to interpret databases and queries in the semantics. As usual, we introduce the notion of realization of a database.

DEFINITION 4.25 (Realization). Given a database $\Gamma$, and a set of resources $\gamma$, an **S**-*realization* of $(\Gamma, \gamma)$ in an **S**-structure $M = (W, \circ, \leq, 0, V)$, is a mapping

$$\rho : \mathcal{A} \rightarrow W$$

such that:

1. $\rho(\mathbf{v_0}) = 0$;
2. if $y : B \in \Gamma$ then $M, \rho(y) \models B$.

In order to define the notion of validity of a query, we need to introduce some further notation. Given an **S**-realization $\rho$, $\gamma$ and $x$, we define

$\rho(\gamma) = 0$ if $\gamma = \emptyset$,
$\rho(\gamma) = \rho(x_1) \circ \ldots \circ \rho(x_n)$ if $\gamma = \{x_1, \ldots, x_n\}$, where $x_1 < \ldots < x_n$
$\rho(< \gamma, x >) = \rho(\gamma)$ if $x \in \gamma$,
$\rho(< \gamma, x >) = \rho(\gamma) \circ 0$ if $x \notin \gamma$.

Moreover, if $\gamma = \{x_1, \ldots, x_n\}$, $\delta = \{y_1, \ldots, y_m\}$ (ordered as shown) we write

$$\rho(\gamma) \circ \rho(\delta) = \rho(x_1) \circ \ldots \circ \rho(x_n) \circ (\rho(y_1) \circ \ldots \circ \rho(y_m)).$$

DEFINITION 4.26 (Valid query). Let $Q = \Gamma, \gamma \vdash^? x : A$ be a **S**-regular query, we say that $Q$ is **S**-*valid* if for every **S**-structure $M$, for every realization $\rho$ of $\Gamma$ in $M$, we have

$$M, \rho(< \gamma, x >) \models A.$$

According to the definition above, the **S**-validity of the query $\vdash^? \mathbf{v_0} : A$ means that the formula $A$ is **S**-valid (i.e. $\models^{Fine}_{\mathbf{S}} A$).

To prove the soundness of the proof systems, we need to relate the conditions involved in the reduction rule to the algebraic properties of $\circ$. This is the purpose of the following lemma.

LEMMA 4.27. *Given an* **S**-*regular query* $\Delta, \delta \vdash^? x : G$, *let* $\delta_i$ *and* $x_i$, *for* $i = 0, \ldots, k$, *such that*

1. $\delta_0 = \{x_0\}$, *for some* $x_0 \in \delta$,
2. $\bigcup_{i=0}^{k} \delta_i = \delta$,
3. $Red^{\mathbf{S}}(\delta_0, \ldots, \delta_k, x_0, \ldots, x_k; x)$ .

*For every* **S***-structure* $M$ *and every realization* $\rho$ *of* $(\Delta, \delta)$ *in* $M$*, we have*

$$\rho(< \delta_0, x_0 >) \circ \rho(< \delta_1, x_1 >) \circ \ldots \circ \rho(< \delta_k, x_k >) \leq \rho(< \delta, x >).$$

THEOREM 4.28. *Let* $Q = \Gamma, \gamma \vdash^? x : A$ *be an* **S***-regular query, if* $Q$ *succeeds in the proof system for* **S** *then* $Q$ *is* **S***-valid. In particular, if* $\mathbf{v_0} : A$ *succeeds in the proof system for* **S***, then* $\models^{Fine}_{\mathbf{S}} A$.

**Proof.** By induction on the height $h$ of a successful derivation of $Q$.

- If $h = 0$, then the query immediately succeeds, $x : q \in \Gamma$ and $Succ^{\mathbf{S}}(\gamma, x)$ holds. Let $M$ be an **S**-structure and $\rho$ an **S**- realization of $\Gamma$ in $M$. We have $M, \rho(x) \models q$. In the case of **BCK**, we have that $x \in \gamma$; it easily seen that $\rho(x) \leq \rho(< \gamma, x >)$, whence the result follows by the hereditary property; in all the other cases, we have $\gamma = \{x\}$, thus the claim follows by $\rho(x) = \rho(< \gamma, x >)$.

- Let $h > 0$ and the implication rule is applied to derive $Q$, that is, $A = B \rightarrow C$ and from $Q$, we step to $Q'$:

$$Q' = \Gamma \cup \{z : B\}, \gamma \cup \{z\} \vdash^? z : C,$$

  where $z > \max(Lab(\Gamma))$. $Q'$ succeeds by a shorter derivation. Suppose that $Q$ is not **S**-valid, let $M = (W, \circ, 0, V)$ be an **S**-structure and $\rho$ a realization of $\Gamma$ such that $M, \rho(< \gamma, x >) \not\models B \rightarrow C$. Let $\rho(< \gamma, x >) = a$, then there is some $b \in W$, such that $M, b \models B$, but $M, a \circ b \not\models C$. Let $Q' = \Gamma' = \Gamma \cup \{z : C\}, \gamma' = \gamma \cup \{z\}$. By the induction hypothesis $Q'$ is **S**-valid. We can define a realization $\rho'$ for $\Gamma'$, by letting $\rho'(z) = b$, and $\rho'(u) = \rho(u)$, for $u \neq z$. We have

$$\rho'(\gamma') = \rho(\gamma) \circ \rho'(z) = a \circ b,$$

  whence

$$M, \rho'(\gamma') \not\models C,$$

  which contradicts the induction hypothesis on $Q'$.

- Let $h > 0$ and suppose the first step in a derivation of $Q$ is by reduction. Then $A$ is an atom $q$, there is $z : C \in \Gamma$, with $C : D_1 \rightarrow \ldots \rightarrow D_k \rightarrow q$, and from $Q$ we step to

$$Q_i : \Gamma, \gamma_i \vdash^? x_i : D_i,$$

  for some $\delta_i$ and $x_i$ (for $i = 0, \ldots, k$) such that:

  1. $x_0 = z, \gamma_0 = \{z\}$, and $\bigcup_{i=0}^k \gamma_i = \gamma$.
  2. $Red^{\mathbf{S}}(\gamma_0, \ldots, \gamma_k, x_0, \ldots, x_k; x)$.

Let $M$ be an **S**-structure and $\rho$ be a realization of $\Gamma$ in $M$. We show that $M, \rho(<\gamma, x>) \models q$ holds. Since each $Q_i$ succeeds by a shorter derivation, we get (by the induction hypothesis)

$$M, \rho(<\gamma_i, x_i>) \models D_i.$$

On the other hand, being $\rho$ a realization of $\Gamma$ we have:

$$M, \rho(<\gamma_0, x_0>) \models D_1 \to \ldots \to D_k \to q.$$

Thus, we obtain

$$M, \rho(<\gamma_0, x_0>) \circ \rho(<\gamma_1, x_1>) \circ \ldots \circ \rho(<\gamma_k, x_k>) \models q.$$

By Lemma 4.27, we have

$$\rho(<\gamma_0, x_0>) \circ \rho(<\gamma_1, x_1>) \circ \ldots \circ \rho(\gamma_k, x_k>) \leq \rho(<\gamma, x>),$$

So that, by the hereditary property we obtain:

$$M, \rho(<\gamma, x>) \models q. \qquad \blacksquare$$

## 3 EXTENDING THE LANGUAGE

In this section we show how we can extend the language by some other connectives. We allow extensional conjunction ($\wedge$), disjunction ($\vee$), and intensional conjunction or *tensor* ($\otimes$). The distinction between $\wedge$ and $\otimes$ is typical of substructural logics and it comes from the rejection of some structural rule: $\wedge$ is the usual lattice-inf connective, $\otimes$ is close to a residual operator with respect to $\to$. In relevant logic literature $\otimes$ is often called *fusion* or *cotenability* and denoted by $\circ$.[10] The addition of the above connectives presents some semantic options. The most important one is whether *distribition* (dist) of $\wedge$ and $\vee$ is assumed or not. The list of axioms/rules below characterizes distributive substructural logics.

DEFINITION 4.29 (Axioms for $\wedge, \otimes, \vee$).

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

[10] We follow here the terminology and notation of linear logic [Girard, 1987].

7. $\dfrac{A \quad B}{A \wedge B}$

8. $\dfrac{A \rightarrow B \rightarrow C}{A \otimes B \rightarrow C}$

9. $\dfrac{A \otimes B \rightarrow C}{A \rightarrow B \rightarrow C}$

(e-$\wedge$) For **E** and **E-W** only

$$\Box A \wedge \Box B \rightarrow \Box(A \wedge B)$$

where $\Box C =_{def} (C \rightarrow C) \rightarrow C$.

(dist) $A \wedge (B \vee C) \rightarrow (A \wedge B) \vee C$.

As we have said, the addition of distribution (dist) is a semantic choice, which may be argued. However, for the fragment of the language we consider in this section it does not really matter whether distribution is assumed or not. This fragment roughly corresponds to the Harrop fragment of the previous chapter (see Section 4); since we do not allow positive occurrences of disjunction, the presence of distribution is immaterial.[11] We have included distribution to have a complete axiomatization with respect to the semantics we will adopt.

As in the case of modal logics (see Section 4), we distinguish D-formulas, which are the constituents of databases, and G-formulas which can be asked as goals.

DEFINITION 4.30. Let D-formulas and G-formulas be defined as follows:

$D := q \mid G \rightarrow D$,
$CD := D \mid CD \wedge CD$,
$G := q \mid G \wedge G \mid G \vee G \mid G \otimes G \mid CD \rightarrow G$.

A database $\Delta$ is a finite set of *labelled* of D-formulas.

A database corresponds to a $\otimes$-composition of conjunctions of D-formulas. Formulas with the same label are thought as $\wedge$-conjuncted. Every D-formula has the form

$G_1 \rightarrow \ldots \rightarrow G_k \rightarrow q$,

In the systems **R**, **L**, **BCK**, we have the theorems

$$(A \rightarrow B \rightarrow C) \rightarrow (A \otimes B \rightarrow C) \quad \text{and} \quad (A \otimes B \rightarrow C) \rightarrow (A \rightarrow B \rightarrow C)$$

Thus, in these systems we can simplify the syntax of (non atomic) D-formulas to $G \rightarrow q$ rather than $G \rightarrow D$. This simplification is not allowed in the other systems where we only have the weaker relation

[11] In the fragment we consider we trivially have (for any **S**) $\Gamma, \alpha \vdash^? x : A \wedge (B \vee C)$ implies $\Gamma, \alpha \vdash^? x : A \vee (B \wedge C)$, where $\Gamma$ is a set of D-formulas and $A, B, C$ are G-formulas (see below).

$$\vdash (A \to B \to C) \;\Leftrightarrow\; \vdash A \otimes B \to C.$$

The extent of Definition 4.30 is shown by the following propositions.

PROPOSITION 4.31. *Every formula on ($\wedge, \vee, \to, \otimes$) without*

- *positive*[12] *(negative) occurrences of $\otimes$ and $\vee$ and*
- *occurrences of $\otimes$ within a negative (positive) occurrence of $\wedge$*

*is equivalent to a $\wedge$-conjunction of D-formulas (G-formulas).*

The reason we have put the restriction on nested occurrences of $\otimes$ within $\wedge$ is that, on the one hand, we want to keep the simple labelling mechanism we have used for the implicational fragment, and on the other we want to identify a common fragment for all systems to which the computation rules are easily extended. The labelling mechanism no longer works if we relax this restriction. For instance, how could we handle $A \wedge (B \otimes C)$ as a D-formula? We should add $x : A$ and $x : B \otimes C$ in the database. The formula $x : B \otimes C$ cannot be decomposed, unless we use complex labels: intuitively we should split $x$ into some $y$ and $z$, add $y : B$ and $z : C$, and remember that $x, y, z$ are connected (in terms of Fine semantics the connection would be expressed as $x = y \circ z$). We prefer to accept the above limitation and postpone the investigation of a broader fragment to future work.[13]

The computation rules can be extended to this fragment without great effort.

DEFINITION 4.32 (Proof system for the extended language). We give the rules for queries of the form

$$\Delta, \delta \;\vdash^{?}\; x : G,$$

where $\Delta$ is a set of D-formulas and $G$ is a G-formula.

- (success)

  $$\Delta, \delta \;\vdash^{?}\; x : q \text{ succeeds},$$

  if $x; q \in \Delta$ and $Succ^{\mathbf{S}}(\delta, x)$.

- (implication) from

  $$\Delta, \delta \;\vdash^{?}\; x : CD \to G$$

[12]Positive and negative occurrences are defined as follows: $A$ occurs positively in $A$; if $B\#C$ occurs positively (negatively) in $A$ (where $\# \in \{\wedge, \vee, \otimes\}$), then $B$ and $C$ occur positively (negatively) in $A$; if $B \to C$ occurs positively (negatively) in $A$, then $B$ occurs negatively (positively) in $A$ and $C$ occurs positively (negatively) in n $A$. We say that a connective $\#$ has a positive (negative) occurrence in a formula $A$ if there is a formula $B\#C$ which occurs positively (negatively) in $A$.

[13]In some logics, such as **L**, we do not need this restriction since we have the following property: $\Gamma, A \wedge B \vdash C$ implies $\Gamma, A \vdash C$ or $\Gamma, B \vdash C$. Thus, we can avoid introducing extensional conjunctions into the database, and instead introduce only one of the two conjuncts (at choice). This approach is followed by Harland and Pym [1991]. However the above property does not hold for **R** and other logics.

if $CD = D_1 \wedge \ldots \wedge D_n$, we step to

$$\Delta \cup \{y : D_1, \ldots, y : D_n\}, \delta \cup \{y\} \ \vdash^? \ y : G$$

where $y > \max(Lab(\Delta))$, (hence $y \notin Lab(\Delta)$).

- (reduction) from

$$\Delta, \delta \ \vdash^? \ x : q$$

if there is $z : G_1 \rightarrow \ldots \rightarrow G_k \rightarrow q \in \Delta$ and there are $\delta_i$, and $x_i$ for $i = 0, \ldots, k$ such that:

1. $\delta_0 = \{z\}$, $x_0 = z$,
2. $\bigcup_{i=0}^{k} \delta_i = \delta$,
3. $Red^{\mathsf{S}}(\delta_0, \ldots, \delta_k, x_0, \ldots, x_k; x)$,

then for $i = 1, \ldots k$, we step to

$$\Delta, \delta_i, \ \vdash^? \ x_i : G_i.$$

- (conjunction) from

$$\Delta, \delta \ \vdash^? \ x : G_1 \wedge G_2$$

step to

$\Delta, \delta \ \vdash^? \ x : G_1$ and $\Delta, \delta \ \vdash^? \ x : G_2$.

- (disjunction) from

$$\Delta, \delta \ \vdash^? \ x : G_1 \vee G_2$$

step to

$\Delta, \delta \ \vdash^? \ x : G_i$ for $i = 1$ or $i = 2$.

- (tensor) from

$$\Delta, \delta \ \vdash^? \ x : G_1 \otimes G_2$$

if there are $\delta_1$, $\delta_2$, $x_1$ and $x_2$ such that

1. $\delta = \delta_1 \cup \delta_2$,
2. $Red^{\mathsf{S}}(\delta_1, \delta_2, x_1, x_2; x)$,

step to

$\Delta, \delta_1 \ \vdash^? \ x_1 : G_1$ and $\Delta, \delta_2 \ \vdash^? \ x_2 : G_2$.

An easy extension of the method is the addition of the truth constants $\mathbf{t}$, and $\top$ which are governed by the following axioms/rules

$$A \to \top,$$
$$\vdash \mathbf{t} \to A \text{ iff} \vdash A.$$

Plus the axiom of $(\mathbf{t} \to A) \to A$ for **E** and **E-W**. We can think of $\mathbf{t}$ as defined by propositional quantification

$$\mathbf{t} =_{def} \forall p(p \to p).$$

Equivalently, given any formula $A$, we can assume that $\mathbf{t}$ is the conjunction of all $p \to p$ such that the atom $p$ occurs in $A$. Basing on this definition, it is not difficult to handle $\mathbf{t}$ in the goal-directed way and we leave to the reader to work out the rules. The treatment of $\top$ is straightforward.

EXAMPLE 4.33. Let $\Delta$ be the following database:

$$x_1 : e \wedge g \to d,$$
$$x_2 : (c \to d) \otimes (a \vee b) \to p,$$
$$x_3 : c \to e,$$
$$x_3 : c \to g,$$
$$x_4 : (c \to g) \to b.$$

In Figure 4.2, we show a successful derivation of

$$\Delta, \{x_1, x_2, x_3, x_4\} \vdash^? x_4 : p$$

in relevant logic **E** (and stronger systems). We leave to the reader to justify the steps according to the rules. The success of this query corresponds to the validity of the following formula in **E**:

$$\begin{gathered}\big[(e \wedge g \to d) \otimes ((c \to d) \otimes (a \vee b) \to p) \otimes ((c \to e) \wedge \\ \wedge (c \to g)) \otimes ((c \to g) \to b)\big] \to p.\end{gathered}$$

$$
\begin{array}{ccc}
 & \Delta, \{x_1, x_2, x_3, x_4\} \vdash^? x_4 : p & \\
 & | & \\
 & \Delta, \{x_1, x_3, x_4\} \vdash^? x_4 : (c \to d) \otimes (a \vee b) & \\
\swarrow & & \searrow \\
\Delta, \{x_1, x_3\} \vdash^? x_3 : c \to d & & \Delta, \{x_3, x_4\} \vdash^? x_4 : a \vee b \\
| & & | \\
\Delta, x_5 : c, \{x_1, x_3, x_5\} \vdash^? x_5 : d & & \Delta, \{x_3, x_4\} \vdash^? x_4 : b \\
| & & | \\
\Delta, x_5 : c, \{x_3, x_5\} \vdash^? x_5 : e \wedge g & & \Delta, \{x_3\} \vdash^? x_4 : c \to g \\
\swarrow \quad \searrow & & | \\
\Delta, x_5 : c, \{x_3, x_5\} \vdash^? x_5 : e \qquad \Delta, x_5 : c, \{x_3, x_5\} \vdash^? x_5 : g & & \Delta, x_6 : c, \{x_3, x_6\} \vdash^? x_6 : g \\
| \qquad\qquad\qquad | & & | \\
\Delta, x_5 : c\{x_5\} \vdash^? x_5 : c \qquad \Delta, x_5 : c, \{x_5\} \vdash^? x_5 : c & & \Delta, x_6 : c, \{x_6\} \vdash^? x_6 : c
\end{array}
$$

Figure 4.2. Derivation for Example 4.33.

## *Routley–Meyer Semantics, Soundness and Completeness*

We turn to prove the soundness and the completeness of the procedure for the extended language. In the previous section we introduced a semantics for substructural implicational logics and we have proved the soundness and completness with respect to it. We would like to extend this correspondence to the Harrop fragment of this section. We first extend Definition 4.22 by giving the truth conditions for the additional connectives.

DEFINITION 4.34. Let $M = (W, \circ, 0, V)$ be a **S**-structure, let $a \in W$ we stipulate:

> $M, a \models A \wedge B$ iff $M, a \models A$ and $M, a \models B$,
> $M, a \models A \vee B$ iff $M, a \models A$ or $M, a \models B$,
> $M, a \models A \otimes B$ iff there are $b, c \in W$, s.t. $b \circ c \leq a$ and $M, a \models A$ and $M, a \models B$.

It is straightforward to extend Theorem 4.28 obtaining the soundness of the proof procedure with respect to this semantics.

THEOREM 4.35. *Let* $\Gamma, \gamma \vdash^? x : G$ *succeeds in the system* **S**, $\gamma = \{x_1, \ldots, x_k\}$ *(ordered as shown) and* $i = 1, \ldots, k$ *let* $S_i = \{A \mid x_i : A \in \Gamma\}$*; then*

$$\models^{Fine}_{\mathbf{S}} (\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \to G.$$

What about the completeness? We will show that it holds for this fragment although we will prove it rather indirectly. We first need to clarify and discuss the basic problem of this semantics. A remarkable point is that the semantics of Definitions 4.22 and 4.34 does *not* match with the axiomatization given at the beginning of this section, namely it is too strong. In the case of **R**, **E**, **T** (i.e. the systems with contraction) it makes valid the formula

$$(A \to B \vee C) \wedge (B \to C) \to (A \to C).$$

This formula is not a theorem of **R**, whence of the other mentioned systems. This means that one cannot extend the semantics of Section 4.22 with the simple boolean definition of conjunction and disjunction. Historically, this problem has been noticed first by Urquhart for his semi-lattice semantics and has lead to the formulation of many alternative but related semantics for susbtractural logics (see the references at the beginning of Section 2.2). To this regard the definition of **S**-structures given in 4.22 is a simplification of the original one by Fine, which works for the full propositional language. In the original definition, Fine structures are enriched by a subset of special worlds correponding to 'prime' (or saturated) theories; these special worlds are needed to interprete disjunctive formulas, they 'decide' disjunctions.

Instead of making a distinction between prime and non-prime worlds, there is another possibility: generalize the semantics by considering a ternary relation rather than a composition operation $\circ$ as in Fine structures. This options leads to the semantics by Routley and Meyer [1982], that is now considered as the standard semantics for relevant and substructural logics with distribution. In Routley–Meyer structures (RM structures for short) worlds are equipped by a three-place accessibility relation $Rabc$. If we interpret $a, b, c$ as pieces of information, one can read $Rabc$ as something of the sort 'the combination of $a$ and $b$ is included in $c$'. With this interpretation, there is an obvious connection with the Fine structures: given $a, b, c \in W$, we can define the ternary relation $a \circ b \leq c$. Thus, every **S**-structure determines a RM structure, but not *vice versa*.[14]

Our plan is the following: we introduce RM-semantics and we prove the completeness with respect to it. Then we show that for the Harrop fragment we consider, the RM-semantics and the semantics of Definitions 4.22 and 4.34 are equivalent. This essentially means that, as far as *positive* disjunctions are not allowed in database-formulas, the simplified Fine semantics of Section 4.22 extended by the obvious truth-definitions of conjunction and disjunction is equivalent to the ternary-relation semantics by Routley and Meyer.[15] The reason why we prove the completeness with-respect to the RM-semantics is that it is easier to make a canonical structure construction with respect to this semantics, rather than to Fine's. Moreover, the correspondence of our deduction procedure with the Routley–Meyer

[14] Another reading of $Rabc$ is relative inclusion: $b \leq_a c$ '$b$ is included in $c$, from the point of view of $a$'.

[15] We recall once more, that our language does not contain negation; if negation is present this equivalence does no longer hold.

semantics might be of interest in itself. In the canonical structure the ternary accessibility relation happens to be defined directly by the constraints involved in the reduction rule.

DEFINITION 4.36. Let us fix a language $\mathcal{L}$, a RM **S**-structure $M$ for $\mathcal{L}$ is a tuple of the form:

$$M = (W, R, 0, V),$$

where $W$ is a non-empty set, $R$ is a ternary relation on $W$, $0 \in W$, $V$ is a function of type $W \to Pow(Var)$. We define the relation $\leq$ by stipulating:

$$a \leq b \;\equiv\; R0ab.$$

In all structures the following properties are assumed to hold:

(i) $\leq$ is transitive and reflexive.

(ii) $a' \leq a$ and $Rabc$ implies $Ra'bc$
$c \leq c'$ and $Rabc$ implies $Rabc'$.

(iii) $a \leq b$ implies $V(a) \subseteq V(b)$.

The structures may satisfy additional properties. To formulate them we introduce the following notation:

$$R^2(ab)cd \;\equiv\; \exists e(Rabe \wedge Recd)$$
$$R^2a(bc)d \;\equiv\; \exists e(Raed \wedge Rbce).$$

The properties are the following:

(rm1) $R^2(ab)cd \;\to\; R^2a(bc)d$

(rm2) $R^2(ab)cd \;\to\; R^2b(ac)d$

(rm3) $Rabc \;\to\; R^2(ab)bc$

(rm4) $\forall a \exists b(Raba \wedge \forall d \forall e(Rbde \to R0de))$

(rm5) $Rabc \;\to\; Rbac.$

(rm6) $R00a.$

**Truth conditions** for $a \in W$, we define

- $M, a \models p$ if $p \in V(a)$;
- $M, a \models A \to B$ if

$$\forall b, c \in W(Rabc \wedge M, b \models A \;\Rightarrow\; M, c \models B).$$

- $M, a \models A \wedge B$ iff $M, a \models A$ and $M, a \models B$;

Table 4.4. Specific conditions for RM semantics.

| Logic | (rm1) | (rm2) | (rm3) | (rm4) | (rm5) | (rm6) |
|---|---|---|---|---|---|---|
| **FL** | * | | | | | |
| **T-W** | * | * | | | | |
| **T** | * | * | * | | | |
| **E-W** | * | * | | * | | |
| **E** | * | * | * | * | | |
| **L** | * | * | | * | * | |
| **R** | * | * | * | * | * | |
| **BCK** | * | * | | * | * | * |

- $M, a \models A \otimes B$ iff there are $b, c \in W$, such that $Rbca$ and $M, b \models A$ and $M, c \models B$;
- $M, a \models A \vee B$ iff $M, a \models A$ or $M, a \models B$.

We say that $A$ is *valid* in $M$ (denoted by $M \models A$) if $M, 0 \models A$. We say that $A$ is (RM)**S**-valid if it is valid in all RM **S**-structures; validity is denoted by $\models_{\mathbf{S}}^{RM} A$.

The mapping between logics **S** and (sets of) conditions is shown in Table 5.4.

Notice that each condition (rm**i**) is closely related to the corresponding condition (a**i**) of Definition 4.22. Perhaps the only one that is not apparent is the correspondence of (a4) and (rm4). The transcription of (a4) would be $Ra0a$. But this is too strong as argued in [Anderson *et al.*, 1992], page 172. (rm4) says that for each $a$ there is a $b$ (which may depend on $a$) which behaves as a right identity.

The completeness is proved by a canonical model construction.

DEFINITION 4.37. Let $M = (W, R, 0, V)$ be the **S**-structure defined as follows:

- $0 = (\emptyset, \mathbf{v_0})$,
- $W$ is the set of pairs $(\Gamma, x)$ (including $0$) such that $\Gamma$ is a database and $x$ is a label, with the following restrictions (confront the definition of regular query, Definition 4.8):
  - for **R**, **L**, **BCK**, if $\Gamma \neq \emptyset$, then $x \in Lab(\Gamma)$,
  - for **T**, **T-W**, **FL**, $x = \max(Lab(\Gamma))$ (thus, if $\Gamma = \emptyset$, then $x = \mathbf{v_0}$),
  - for **E**, **E-W**, $x \geq \max(Lab(\Gamma))$.
- The relation $R$

  $$R(\Gamma, x)(\Delta, y)(\Sigma, z)$$

  holds iff the following conditions (1) and (2) (or (1) and (2′) for **BCK**) hold

  (1) $Red^{\mathbf{S}}(\lambda_\Gamma, \lambda_\Delta, x, y; z)$

(2) $\Gamma \cup \Delta = \Sigma$ for all systems, but **BCK**,

(2′) $\Gamma \cup \Delta \subseteq \Sigma$ for **BCK**.

- $V(\Gamma, x) = \{p :$ atom $\mid \Gamma, \lambda_\Gamma \ \vdash^? \ x : p$ succeeds$\}$.

Notice that the ternary relation is defined as follows: let $\alpha, \beta, \gamma$ be sets of labels, then we have $R\alpha\beta\gamma$ holds if and only if the following conditions hold

- for **R**, $\alpha \cup \beta = \gamma$,
- for **L**, $\alpha \cup \beta = \gamma$ and $\alpha \cap \beta = \emptyset$,
- for **T**, $\alpha \cup \beta = \gamma$ and $\max(\alpha) \leq \max(\beta)$,
- for **FL**, $\alpha \cup \beta = \gamma$ and $\max(\alpha) < \min(\beta)$,
- for **BCK**, $\alpha \cup \beta \subseteq \gamma$ and $\alpha \cap \beta = \emptyset$.

For **E** and **E-W** we must consider pairs $(\delta, u)$, where $\max(\delta) \leq u$, and we have $R(\alpha, x)(\beta, y)(\gamma, z)$ iff

$\alpha \cup \beta = \gamma$ and $x \leq y = z$.

This definition of the ternary relation essentially summarizes the restrictions on Modus Ponens in the natural deduction formulation [Anderson and Belnap, 1975], where MP takes the form:

$$\frac{\alpha : A \to B \quad \beta : A}{\gamma : B}$$

if $R\alpha\beta\gamma$ holds.

We show that the structure $M$ we have just defined satisfies all properties of Definition 4.37.

PROPOSITION 4.38. *The relation* $(\Gamma, x) \leq (\Delta, y) \ \equiv \ R0(\Gamma, x)(\Delta, y)$ *satisfies the conditions (i)–(iii) of Definition 4.37.*

PROPOSITION 4.39. *For each system* **S**, *the relation R satisfies the specific conditions for* **S**.

PROPOSITION 4.40. *For each* $(\Gamma, x) \in W$, *and goal* $G$, *we have:*

- *(a)* $M, (\Gamma, x) \models G$ *iff* $\Gamma, \lambda_\Gamma \ \vdash^? \ x : G$ *succeeds.*
- *(b) If* $\Delta = \{u : D_1, \ldots, u : D_m\}$, *then* $M, (\Delta, u) \models \bigwedge_{i=1}^m D_i$.

THEOREM 4.41 (Completeness). *Let* $\Gamma$ *be a labelled set of D-formulas with labels* $\lambda_\Gamma = \{x_1, \ldots, x_k\}$ *ordered as shown. Let* $S_i = \{D \mid x_i : D \in \Gamma\}$ *for* $i = 1, \ldots, k$, $x = \max(\lambda_\Gamma)$, *and let* $G$ *be a G-formula, then we have* *if* $\models^{RM}_{\mathbf{S}} (\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \rightarrow G$ *(equivalently* $\models^{RM}_{\mathbf{S}} \bigwedge S_1 \rightarrow \ldots \rightarrow \bigwedge S_k \rightarrow G$ *), then* $\Gamma, \lambda_\Gamma \ \vdash^? \ x : G$ *succeeds.*

**Proof.** By contraposition, suppose that $\Gamma, \lambda_\Gamma \ \vdash^? \ x : G$ does not succeed. Then by Proposition 4.40, we have

$$M, (\Gamma, x) \not\models G.$$

Let us consider the following databases $\Gamma_i \subseteq \Gamma$ corresponding to $S_i$:

$$\Gamma_1 = \{x_1 : D \mid x_1 : D \in S_1\}, \ldots, \Gamma_k = \{x_k : D \mid D \in S_k\},$$

then we have: $(\Gamma_i, x_i) \in W$, for $i = 1, \ldots, k$. By Lemma 4.40, we also have

$$M, (\Gamma_i, x_i) \models D \text{ for each } D \in S_i \text{ and for } i = 1, \ldots, k,$$

whence

(*) $\quad M, (\Gamma_i, x_i) \models \bigwedge S_i$, for $i = 1, \ldots, k$.

It is easy to see that $Red^{\mathbf{S}}(\gamma_{\Gamma_1}, \gamma_{\Gamma_2}, x_1, x_2; x_2)$ holds and that $(\Gamma_1 \cup \Gamma_2, x_2) \in W$, so that also

$$R(\Gamma_1, x_1)(\Gamma_2, x_2)(\Gamma_1 \cup \Gamma_2, x_2) \text{ holds.}$$

By (*), we can conclude that

$$M, (\Gamma_1 \cup \Gamma_2, x_2) \models \bigwedge S_1 \otimes \bigwedge S_2.$$

We can repeat the same argument and show that

$$R(\Gamma_1 \cup \Gamma_2, x_2)(\Gamma_3, x_3)(\Gamma_1 \cup \Gamma_2 \cup \Gamma_3, x_3) \text{ holds,}$$

so that we get $M, (\Gamma_1 \cup \Gamma_2 \cup \Gamma_3, x_3) \models \bigwedge S_1 \otimes \bigwedge S_2 \otimes \bigwedge S_3$ holds. Proceeding in this way, we finally get

2. $M, (\Gamma, x) \models \bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k$.

Since $R(\emptyset, \mathbf{v_0})(\Gamma, x)(\Gamma, x)$, by 1. and 2., we get

$$M, 0 \not\models (\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \rightarrow G,$$

which completes the proof. ■

We finally show that for G- and D-formulas RM-semantics is equivalent to Fine semantics of Definitions 4.22 and 4.34. We show that any RM structure determines a Fine structure essentially equivalent as far as D- and G-formulas are concerned. The construction is rather standard (cfr. [Anderson *et al.*, 1992] page 223) Let $M = (W, R, 0, V)$ be a RM-structure. Call a subset $X \subseteq W$ *upward-closed* if

$a \in X$ and $a \leq b$ (i.e. $R0ab$) implies $b \in X$. We define the Fine-structure $\mathbf{S}_{\mathrm{M}} = (\mathbf{W}_{\mathrm{M}}, \circ_{\mathrm{M}}, \mathbf{0}_{\mathrm{M}}, \leq, \mathbf{V}_{\mathrm{M}})$, by stipulating

$$\begin{array}{rcl} \mathbf{W}_{\mathrm{M}} & = & \{X \subseteq W \mid X \text{ is upward closed}\}, \\ \mathbf{0}_{\mathrm{M}} & = & \{a \in W \mid \forall bc \in W(Rabc \rightarrow R0bc)\}, \\ X \circ_{\mathrm{M}} Y & = & \{c \in W \mid \exists a \in X\ \exists b \in Y\, Rabc\}, \\ X \leq Y & \Leftrightarrow & Y \subseteq X, \\ \mathbf{V}_{\mathrm{M}}(X) & = & \bigcap_{a \in X} V(a). \end{array}$$

Notice that $\circ_{\mathrm{M}}$ and $\mathbf{0}_{\mathrm{M}}$ are well-defined, that is $X \circ_{\mathrm{M}} Y$ is upward-closed, and so is $\mathbf{0}_{\mathrm{M}}$. The reader is invited to check the following.

PROPOSITION 4.42. *(i) $\mathbf{S}_{\mathrm{M}}$ satisfies all the conditions of definition 4.37 common to all structures. (ii) If $M$ satisfy condition (rm**i**) of 4.37, then $\mathbf{S}_{\mathrm{M}}$ satisfies the corresponding condition (a**i**) of 4.22.*

**Proof.** We exemplify (ii) for the case of condition (a4), namely $X \circ_{\mathrm{M}} \mathbf{0}_{\mathrm{M}} \leq X$. The others are left to the reader as an exercise. We have to show

$$X \subseteq X \circ_{\mathrm{M}} \mathbf{0}_{\mathrm{M}}.$$

Let $a \in X$, by condition (rm4), we have that there is $b$ such that $Raba$ and for $\forall c, d \in W\ Rbcd \rightarrow R0cd$. Thus, by definition of $\mathbf{S}_{\mathrm{M}}$, we have $b \in \mathbf{0}_{\mathrm{M}}$ so that $a \in X \circ_{\mathrm{M}} \mathbf{0}_{\mathrm{M}}$. ■

Our purpose is to prove the equivalence of the two semantics for G- and D-formulas. The equivalence actually holds for a class of formulas a bit more general than D- and G-formulas as defined above. Let the class of *extended* G-formulas and D-formulas be defined as follows

$$D := q \mid D \wedge D \mid D \otimes D \mid G \rightarrow D,$$
$$G := q \mid G \wedge G \mid G \otimes G \mid D \rightarrow G \mid G \vee G.$$

The precise relation of the two semantics is made clear in the next proposition.

PROPOSITION 4.43. *Given a RM structure $M = (W, R, 0, V)$ let $\mathbf{S}_{\mathrm{M}} = (\mathbf{W}_{\mathrm{M}}, \circ_{\mathrm{M}}, \mathbf{0}_{\mathrm{M}}, \leq, \mathbf{V}_{\mathrm{M}})$ be the corresponding Fine structure as defined above. For any extended G-formula $G$ and extended D-formula $D$, and $X \in \mathbf{W}_{\mathrm{M}}$*

1. *$\forall a \in X$, $M, a \models D$ implies $\mathbf{S}_{\mathrm{M}}, X \models D$;*
2. *$\mathbf{S}_{\mathrm{M}}, X \models G$ implies $\forall a \in X$, $M, a \models G$.*

**Proof.** The two claims are proved by mutual induction on the complexity of G and D. We exemplify the case of implication and disjunction. Let $D = G \rightarrow D_1$, assume $\forall a \in X$, $M, a \models D$. Let $\mathbf{S}_{\mathrm{M}}, Y \models G$, by the induction hypothesis of 2. we have

(*) $\forall b \in Y, M, b \models G.$

Let $c \in X \circ_{\mathbf{M}} Y$, we have for some $a \in X$ and $b \in Y$, $Rabc$, thus by hypothesis and (*) we get $M, c \models D_1$. We have shown that $\forall c \in X \circ_{\mathbf{M}} Y$, $M, c \models D_1$, thus by the induction hypothesis of 1. we get $\mathbf{S_M}, Y \models D_1$.

Conversely, let $G = D \to G_1$ and $\mathbf{S_M}, X \models D \to G_1$. Let $a \in X$, we show $M, a \models D \to G_1$. To this end let $b, c \in W$, such that $Rabc$ and $M, b \models D$. Define

$$Y = \{u \in W \mid b \leq u\}.$$

By monotony we get $\forall u \in Y,\ M, u \models D$. By the induction hypothesis of 1. we can conclude $\mathbf{S_M}, Y \models D$. Therefore by hypothesis, we have

$$\mathbf{S_M}, X \circ_{\mathbf{M}} Y \models G_1,$$

Since $c \in X \circ_{\mathbf{M}} Y$, by the induction hypothesis of 2., we obtain $M, c \models G_1$.

Let $G = G_1 \vee G_2$, assume $\mathbf{S_M}, X \models G$, then $\mathbf{S_M}, X \models G_1$ or $\mathbf{S_M}, X \models G_2$. Suppose the former holds, then by the induction hypothesis of 2. we get $\forall a \in X$, $M, a \models G_1$, thus also $\forall a \in X,\ M, a \models G_1 \vee G_2$. If $\mathbf{S_M}, X \models G_2$ we argue similarly. ■

Notice that the above proof does not work if disjunction of D-formulas is allowed. For instance, the formula $(A \to B \vee C) \wedge (B \to C) \to (A \to C)$, with $A, B, C$ atoms is not an extended G-formula. From the previous proposition we easily obtain:

THEOREM 4.44. *Let* $\Gamma, \gamma \vdash^? x : G$ *be a query with* $\gamma = \{x_1, \ldots, x_k\}$ *(ordered as shown), and let* $S_i = \{A \mid x_i : A \in \Gamma\}$*,* $i = 1, \ldots, k$*. We have*

$$\models^{Fine}_{\mathbf{S}} (\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \to G$$

*if and only if*

$$\models^{RM}_{\mathbf{S}} (\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \to G.$$

**Proof.** The (only if) part is proved by noticing that a Fine structure $M = (W, \circ, 0, \leq, V)$ determines trivially a RM structure by defining the ternary relation $Rabc \equiv a \circ b \leq c$.

For the (if) part, observe first that $F = (\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \to G$ is an extended G-formula. By the hypothesis $\models^{Fine}_{\mathbf{S}} F$. Let $M$ be any RM-structure and let $\mathbf{S_M}$ be the Fine- structure built from $M$ as described above. By hypothesis $\mathbf{S_M}, \mathbf{0_M} \models F$. By the previous proposition, we get $M, 0 \models F$, as $0 \in \mathbf{0_M}$. ■

To conclude, we submarize all the results of this section in the next theorem.

THEOREM 4.45. *Let* $\Gamma, \gamma \vdash^? x : G$ *be a query with* $\gamma = \{x_1, \ldots, x_k\}$ *(ordered as shown), and let* $S_i = \{A \mid x_i : A \in \Gamma\}$*,* $i = 1, \ldots, k$*. The following are equivalent*

1. $\Gamma, \gamma \vdash^{?} x : G$ *succeeds in the proof systems for* **S**,
2. $(\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \rightarrow G$ *is valid under Fine semantics for* **S**,
3. $(\bigwedge S_1 \otimes \ldots \otimes \bigwedge S_k) \rightarrow G.$ *is valid under RM semantics for* **S**.

# BIBLIOGRAPHY

*

[Anderson and Belnap, 1975] A. R. Anderson and N. D. Belnap. *Entailment, The Logic of Relevance and Necessity*, Vol. 1. Princeton University Press, New Jersey, 1975.

[Anderson *et al.*, 1992] A. R. Anderson, N. D. Belnap and J. M. Dunn. *Entailment, The Logic of Relevance and Necessity*, Vol. 2. Princeton University Press, New Jersey, 1992.

[Bollen, 1991] A. W. Bollen. Relevant logic programming, *Journal of Automated Reasoning*, **7**, 563–585, 1991.

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

[Dershowitz and Manna, 1979] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, **22**, 465-472, 1979.

[Došen, 1988] K. Došen. Sequent systems and grupoid models I. *Studia Logica*, **47**, 353–385, 1988.

[Došen, 1989] K. Došen. Sequent systems and grupoid models II. *Studia Logica*, **48**, 41–65, 1989.

[Došen, 1993] K. Došen. A historical introduction to substructural logics. In *Substructural Logics*, P. Schroeder-Heister and K. Došen, eds. Oxford University Press, 1993.

[Dummett, 1977] M. Dummett. *Elements of Intuitionism*, Oxford University Press, 1977.

[Dunn, 1986] J. M. Dunn. Relevance logic and entailment. In *Handbook of Philosophical Logic*, vol III, D. Gabbay and F. Guenthner, eds. pp. 117–224. D. Reidel, Dordercht, 1986.

[Dyckhoff, 1999] R. Dyckhoff. A deterministic terminating sequent calculi for Gödel–Dummett logic. *Logic Journal of the IGPL*, **7**, 319–326, 1999.

[Eshghi, 1989] K. Eshghi and R. Kowalski. Abduction compared with negation by failure. In *Proceedings of the 6th ICLP*, pp. 234–254, Lisbon, 1989.

[Farinãs, 1986] L. Farinãs del Cerro. MOLOG: a system that extends Prolog with modal logic. *New Generation Computing*, **4**, 35–50, 1986.

[Fine, 1974] K. Fine. Models for entailment, *Journal of Philosophical Logic*, **3**, 347–372, 1974.

[Fitting, 1983] M. Fitting. *Proof methods for Modal and Intuitionistic Logic*, vol 169 of Synthese library, D. Reidel, Dorderecht, 1983.

[Fitting, 1990] M. Fitting. *First-order Logic and Automated Theorem Proving*. Springer-Verlag, 1990.

[Gabbay, 1981] D. M. Gabbay. *Semantical Investigations in Heyting's Intuitionistic Logic*. D. Reidel, Dordrecht, 1981.

[Gabbay, 1985] D. M. Gabbay. $N$-Prolog Part 2. *Journal of Logic Programming*, 251–283, 1985.

[Gabbay, 1987] D. M. Gabbay. Modal and temporal logic programming. In *Temporal Logics and their Applications*, A. Galton, ed. pp/ 197–237. Academic Press, 1987.

[Gabbay, 1991] D. M. Gabbay. Algorithmic proof with diminishing resources. In *Proceedings of CSL'90*, pp. 156–173. LNCS vol 533, Springer-Verlag, 1991.

[Gabbay, 1992] D. M. Gabbay. Elements of algorithmic proof. In *Handbook of Logic in Theoretical Computer Science*, vol 2. S. Abramsky, D. M. Gabbay and T. S. E. Maibaum, eds., pp. 307–408, Oxford University Press, 1992.

[Gabbay, 1996] D. M. Gabbay, *Labelled Deductive Systems* (vol I), Oxford Logic Guides, Oxford University Press, 1996.

[Gabbay and de Queiroz, 1992] D. M. Gabbay and R. De Queiroz. Extending the Curry-Howard interpretation to linear, relevant and other resource logics. *Journal of Symbolic Logic*, **57**, 1319–1365, 1992.

[Gabbay and Kriwaczek, 1991] D. M. Gabbay and F. Kriwaczek. A family of goal-directed theorem-provers based on conjunction and implication. *Journal of Automated Reasoning*, **7**, 511–536, 1991.

[Gabbay and Olivetti, 1997] D. M. Gabbay and N. Olivetti. Algorithmic proof methods and cut elimination for implicational logics - part I, modal implication. *Studia Logica*, **61**, 237–280, 1998.

[Gabbay and Olivetti, 1998] D. Gabbay and N. Olivetti. Goal-directed proof-procedures for intermediate logics. In *Proc. of LD'98 First International Workshop on Labelled Deduction*, Freiburg, 1998.

[Gabbay and Reyle, 1984] D. M. Gabbay and U. Reyle. $N$-Prolog: an Extension of Prolog with hypothetical implications, I. *Journal of Logic Programming*, **4**, 319–355, 1984.

[Gabbay *et al.*, 1999] D. M. Gabbay, N. Olivetti and S. Vorobyov. Goal-directed proof method is optimal for intuitionistic propositional logic. Manuscript, 1999.

[Gallier, 1987] J. H. Gallier. *Logic for Computer Science*, John Wiley, New York, 1987.

[Galmiche and Pym, 1999] D. Galmiche and D. Pym. Proof-Search in type-theoretic languages. To appear in *Theoretical Computer Science*, 1999.

[Giordano *et al.*, 1992] L. Giordano, A. Martelli and G. F. Rossi. Extending Horn clause logic with implication goals. *Theoretical Computer Science*, **95**, 43–74, 1992.

[Giordano and Martelli, 1994] L. Giordano and A. Martelli. Structuring logic programs: a modal approach. *Journal of Logic Programming*, **21**, 59–94, 1994.

[Girard, 1987] J. Y. Girard. Linear logic. *Theoretical Computer Science* **50**, 1–101, 1987.

[Goré, 1998] R. Goré. Substructural logics on display. *Journal of IGPL*, **6**, 451–504, 1998.

[Halpern and Moses, 1990] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of ACM*, **37**, 549–587, 1990.

[Harland and Pym, 1991] J. Harland and D. Pym. The uniform proof-theoretic foundation of linear logic programming. In *Proceedings of the 1991 International Logic Programming Symposium*, San Diego, pp. 304–318, 1991.

[Harland, 1997] J. Harland. On Goal-Directed Provability in Classical Logic, *Computer Languages*, pp. 23:2-4:161-178, 1997.

[Heudering *et al.*, 1996] A. Heudering, M. Seyfried and H. Zimmerman. Efficient loop-check for backward proof search in some non-classical propositional logics. In (P. Miglioli et al. eds.) *Tableaux 96*, pp. 210–225. LNCS 1071, Springer Verlag, 1996.

[Hindley, 1997] J. R. Hindley. *Basic Simple Type Theory*. Cambridge University Press, 1997.

[Hodas and Miller, 1994] J. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, **110**, 327–365, 1994.

[Hudelmaier, 1990] J. Hudelmaier. Decision procedure for propositional $n$- prolog. In (P. Schroeder-Heister ed.) *Extensions of Logic Programming*, pp. 245–251, Springer Verlag, 1990.

[Hudelmaier, 1990] J. Hudelmaier. An $O(n\ log\ n)$-space decision procedure for intuitionistic propositional logic. *Journal of Logic and Computation*, **3**, 63–75, 1993.

[Lambek, 1958] J. Lambek. The mathematics of sentence structure, *American Mathematics Monthly*, **65**, 154–169, 1958.

[Lewis, 1912] C. I. Lewis. Implication and the algebra of logic. *Mind*, **21**, 522–531, 1912.

[Lewis and Langford, 1932] C. I. Lewis and C. H. Langford. *Symbolic Logic*, The Century Co., New York, London, 1932. Second edition, Dover, New York, 1959.

[Lincoln *et al.*, 1991] P. Lincoln, P. Scedrov and N. Shankar. Linearizing intuitionistic implication. In *Proceedings of LICS'91*, G. Kahn ed., pp. 51–62, IEEE, 1991.

[Lloyd, 1984] J. W. Lloyd, *Foundations of Logic Programming*, Springer, Berlin, 1984.

[Loveland, 1991] D. W. Loveland. Near-Horn prolog and beyond. *Journal of Automated Reasoning*, **7**, 1–26, 1991.

[Loveland, 1992] D. W. Loveland. A comparison of three Prolog extensions. *Journal of Logic Programming*, **12**, 25-50, 1992.

[Manna and Pnueli, 1981] Z. Manna and A. Pnueli. Verification of concurrent programs: the temporal framework. In *The Correctness Problem in Computer Science*, Academic Press, London, 1981.

[Martins and Shapiro, 1988] J. P. Martins and S. C. Shapiro. A model for belief revision, *Journal of Artificial Intelligence*, **35**, 25-79, 1988.

[McCarty, 1988a] L. T. McCarty. Clausal intuitionistic logic. I. Fixed-point semantics. *Journal of Logic Programming*, **5**, 1–31, 1988.

[McCarty, 1988b] L. T. McCarty. Clausal intuitionistic logic. II. Tableau proof procedures. *Journal of Logic Programming*, **5**, 93–132, 1988.

[McRobbie and Belnap, 1979] M. A. McRobbie and N. D. Belnap. Relevant analytic tableaux. *Studia Logica*, **38**, 187–200, 1979.

[Miller *et al.*, 1991] D. Miller, G. Nadathur, F. Pfenning and A. Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, **51**, 125–157, 1991.

[Nadathur, 1998] G. Nadathur. Uniform provability in classical logic. *Journal of Logic and Computation*, **8**, 209–229,1998.

[Olivetti, 1995] N. Olivetti. *Algoirthmic Proof-theory for Non-classical and Modal Logics*. PhD Thesis, Dipartimento di Informatica, Universita' di Torino, 1995.

[Ono, 1993] H. Ono. Semantics for substructural logics. In P. Schroeder-Heister and K. Došen(eds.) *Substructural Logics*, pp. 259–291, Oxford University Press, 1993.

[Ono, 1998] H. Ono. Proof-theoretic methods in non-classical logics-an introduction. In (M. Takahashi et al. eds.) *Theories of Types and Proofs*, pp. 267–254, Mathematical Society of Japan, 1998.

[Ono and Komori, 1985] H. Ono and Y. Komori. Logics without the contraction rule. *Journal of Symbolic Logic*, **50**, 169–201, 1985.

[Pnueli, 1981] A. Pnueli. The temporal logic of programs, *Theoretical Computer Science*, **13**, 45–60, 1981.

[Prawitz, 1965] D. Prawitz. *Natural Deduction*. Almqvist & Wiksell, 1965.

[Restall, 1999] G. Restall. *An introduction to substructural logics*. Routledge, to appear, 1999.

[Routley *et al.*, 1982] R. Routley, V. Plumwood, R. K. Meyer, and R. Brady. *Relevant Logic and their Rivals. Part I. The Basic Philosophical and Semantical Theory*. Ridgeview Publishing Company, Atascadero, California, 1982.

[Slaney and Meyer, 1997] G. Slaney and R. Meyer. Logics for two: the semantics of distributive substructural logics. In *Proceedings of Qualitative and Quantitative Practical Reasoning*, pp. 554–567. LNCS 1244, Springer Verlag, 1997.

[Statman, 1979] R. Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, **9**, 67–72, 1979.

[Thistlewaite *et al.*, 1988] P. B. Thistlewaite, M. A. McRobbie and B. K. Meyer. *Automated Theorem Proving in Non Classical Logics*, Pitman, 1988.

[Turner, 1985] R. Turner. *Logics for Artificial Intelligence*, Ellis Horwood Ltd., 1985.

[Troelstra, 1969] A. S. Troelstra. *Principles of Intuitionism*. Springer-Verlag, Berlin, 1969.

[Urquhart, 1972] A. Urquhart. The semantics of relevance logics. *The Journal of Symbolic Logic*, **37**, 159–170, 1972.

[Urquhart, 1984] A. Urquhart. The undecidability of entailment and relevant implication. *The Journal of Symbolic Logic*, **49**, 1059–1073, 1984.

[van Dalen, 1986] D. Van Dalen. Intuitionistic logic. In *Handbook of Philosophical logic, Vol 3*, D. Gabbay and F. Guenther, eds. pp. 225–239. D. Reidel, Dorderecht, 1986.

[Viganò, 1999] L. Viganò. Labelled Non-classical Logics. Kluwer Academic Publishers, 1999, to appear.

[Wallen, 1990] L. A. Wallen, *Automated Deduction in Nonclassical Logics*, MIT Press, 1990.

[Wansing, 1990] H. Wansing. Formulas as types for a hierarchy of sublogics of intuitionistic propositional logic, Technical Report n.9/90, Free University of Berlin, 1990.