SEKI Report

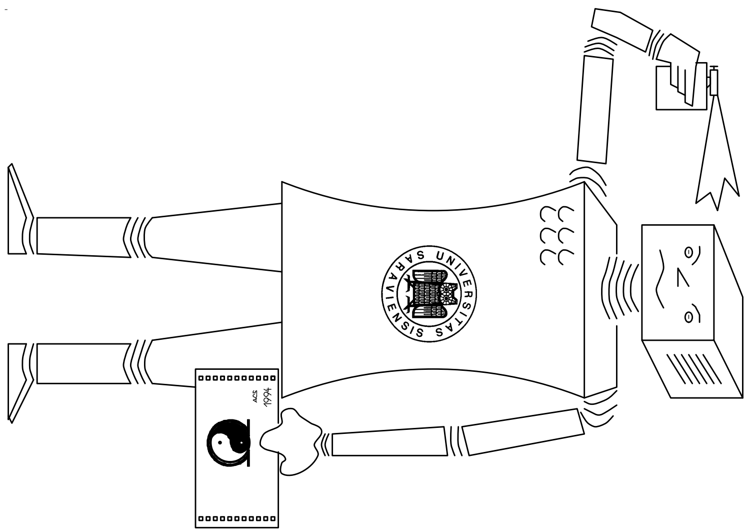
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY

WWW: http://js-sfbsun.cs.uni-sb.de/pub/www/

A Unifying Logical Framework for Reason Maintenance

Detlef Fehrer

SEKI Report SR-96-04



Abstract

We present a way to describe Reason Maintenance Systems using the same formalism for justification based as well as for assumption based approaches. This formalism uses *labelled formulae* and thus is a special case of Gabbay's labelled deductive systems. Since our approach is logic based, we are able to get a *semantics oriented* description of the systems in question.

Instead of restricting ourselves to e.g. propositional Horn formulae, as was done in the past, we admit arbitrary logics. This enables us to characterize systems as a whole, including both the reason maintenance component and the problem solver, nevertheless maintaining a separation between the basic logic and the part that describes the label propagation. The possibility to freely vary the basic logic enables us to not only describe various existing systems, but can help in the design of completely new ones.

We also show, that it is possible to implement systems based directly on our labelled logic and plead for "incremental calculi" crafted to attack undecidable logics.

Furthermore it is shown that the same approach can be used to handle default reasoning, if the propositional labels are upgraded to first order.

Contents

Indeed, even at this stage I predict a time when there will be mathematical investigations of calculi containing contradictions, and people will actually be proud of having emancipated themselves even from consistency.

LUDWIG WITTGENSTEIN

Chapter 1

Introduction

Much effort has been spent on the *automation of reasoning*, and researchers from different background have been working on this topic in various subareas of artificial intelligence and cognitive science. The motivations for doing so can roughly be divided into two main categories:

1. Interest in building (computer) systems that can reason.
2. Interest in a model how humans do it.

Research in computer science and also in artificial intelligence is mainly driven by the first motivation, whereas cognitive science and psychology are usually guided by the second.

Particularly in the field of *artificial intelligence*(AI for short) such systems have been examined. Though the main interest here has certainly also been to implement systems that "reason", many researchers agree that it is worth considering *cognitive models* as well.

This aspect becomes apparent when we consider domains other than mathematics and examine e.g. the phenomena of *commonsense reasoning* For instance mathematical theories are monotonic, i.e. adding further axioms can never defeat a theorem, which means that everything once detected (proven) as true will stay true forever. This is not true in most commonsense reasoning processes. Very often inferences have to be drawn that use premises that later turn out to be false, or worse, sometimes inferences have to be drawn that strictly speaking could not be done, because not all of their premises can be guaranteed to hold (incomplete information) at a given point in time.

Seen as a black box this type of AI system works like some kind of intelligent information storage that is used by other components or directly by man. It is (once and for all or incrementally) provided with *facts* (what is observed or has been told) and later questioned about them. Such a *knowledge base* should, however, not only be able to repeat the items explicitly given to it, but be capable of *deriving* knowledge implicitly contained in the facts given by application of *rules* also supplied or taken from knowledge about the domain or even from common sense. There exists a vast literature on these knowledge bases (see e.g. Frost (1986) and the literature on *deductive databases*, as e.g. Nicolas & Yazdanian (1978), Reiter (1980*a*)).

Hence, in general, any AI system maintains an internal model of the real world (or a suited subset thereof) and/or of its own state of "mind" (beliefs, plans, intentions etc.). It draws inferences and thus adds to the knowledge it was given at the start. Communication with the real world usually runs via *observations* in one direction and possibly some kind of *actions* in the other. Not all of those parts have to be incorporated into every system, but this is the fundamental design.

We shall not talk about the interactions with the real world, as given by observations and actions, in this thesis, but concentrate on the core, which we call the *abstract reasoner*. Here all the communication with the environment is reduced to either getting information through a dedicated input channel or answering questions (called queries) transmitted through a second interface. The questions are all of the same simple type, merely asking whether some proposition holds or not, and the only answers possible are "yes", "no" or possibly "don't know".

As a means to describe what such a system is doing and to formulate its properties, it is often proposed to use some kind of *logic* for the internal representation of knowledge as well as for the reasoning part. This has the advantage that this way programs can be given a declarative semantics. Unfortunately many existing systems simply are not given such a characterization, but programmed in an ad hoc fashion. Even if a logical correspondence is given, easy comparison between different systems tailored to fulfill the same or a related task is not always possible because of the abundance of logical formalisms.

The abstract reasoner can be seen as a "black box" from outside. When we look at the internal behaviour, the simplest realization can be thought of like this: Assuming a logical representation of the knowledge, answering questions is in essence trying to *prove* that the query formula logically follows from the input knowledge, which serves as the axiom set.

Therefore, one way to proceed is to take an ordinary *Automated Theorem Prover* (ATP). In practice, there are, however, several reasons why such a procedure might not be ideal for the task:

1. In contrast to the situation found in Automated Theorem Proving there is not a (nearly) unlimited amount of time available, but tighter time bounds exist. The reason for this is that the answer to one question is generally not the top level goal, but only contributes as a minor part to a bigger task.

2. Again in contrast to ATP a single question can not be considered in isolation. The situation might even occur that the same question is asked several times.

3. The set of axioms is in general not static, but dynamic.

4. In real examples the input knowledge cannot be assumed to be consistent in every case. For many logics this will result in the situation that every question will be answered with "yes", which is certainly not what is desired.

Points 2 and 3 in the preceding list, considered in isolation, do not necessarily object to the ATP-approach of considering every single question as a separate proof. But in connection with point 1 one will certainly try to make use of knowledge obtained during one proof in later ones, thus saving valuable time. So 2 may lead to *lemma generation*. But this in turn contradicts 3, because a derived lemma may not hold in the theory given by the changed set of assumptions. Of course, if the logic is *monotonic* and the axiom set is only increasing, this may not happen. But because of point 4 it may well be the case that items previously given will be retracted, not to speak of real *nonmonotonic inference rules*.

1.1 Reason Maintenance Systems

In order to overcome these difficulties *Reason Maintenance Systems* (RMS) have been developed. Their main task is to maintain a record of the dependencies of items derived. Thus the validity of any item can be traced back to the validity of other items it depends on and so forth.

The general approach pursued in the past has been to view the RMS as an additional component to the problem solver, as shown in figure 1.1. There are numerous different RMSs that have been implemented along this line. All of these use a structure called a *dependency net* (DN). This is a graph, where the nodes represent pieces of information and the edges tell how this information is related. Often the nodes are *labelled*, e.g. with information about whether that node is (currently) believed or not. Almost always there is another kind of nodes, called *justifications*. A justification represents — as the name suggests — the justification for belief in an item. It depends on the state of belief of some antecedent nodes and supports belief in one or several consequent nodes.

RMSs can be divided according to two main criteria:

1. whether they are *justification based* or *assumption based*. In the first case the nodes are labelled according to the current state of belief. So the momentary validity of an item can instantly be checked via lookup of its label. If some item should change its state, e.g. if a premise is retracted, then this information has to be propagated through the graph and the states of the concerned nodes must be changed appropriately. This can take a while.

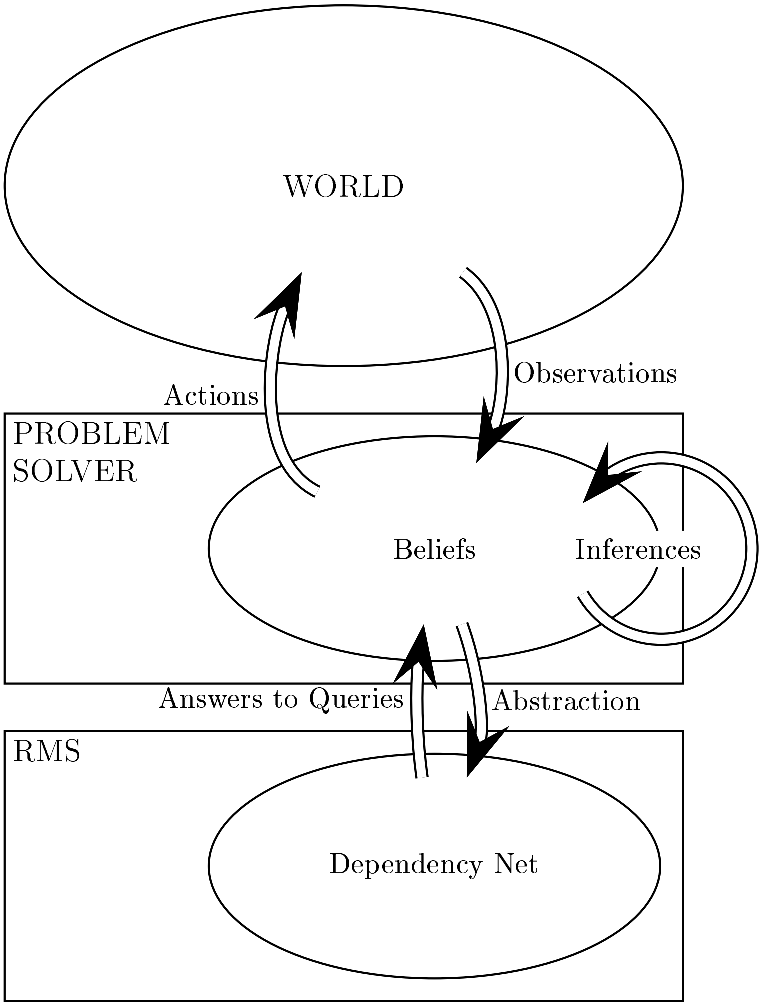


Figure 1.1: The usual RMS approach

In the assumption based approach, the labels at the nodes do not contain simple status information. Instead there exist certain distinguished nodes (the *assumptions*), which are the only ones whose state can be altered arbitrarily. The states of belief for all the other nodes can then be computed for all possible combinations of beliefs in these assumptions. This is done once and for all at the start and written as a label at the nodes. Now the information about the validity of a node given a particular set of assumptions (the RMS terminology is "in a given *context*") can quickly be recovered. The price for this is the costly computation at the beginning.

2. whether they allow for *nonmonotonic justifications* or not. A nonmonotonic justification is a justification that has as an antecedent (possibly among others) the *lack of belief* in an item. This is an interesting possibility, but causes problems because of the nonmonotonicity introduced.

There are two possibilities, why a formerly derived item may have to be retracted. The first is that its justification could be *undermined*, in the sense that the reason for its derivation may have gone. In assumption based approaches this is done by considering contexts where some assumption needed in the proof of the item is missing. In all approaches containing nonmonotonic justifications this can also happen if justifications are added for some items which were formerly assumed to be not believed.

The second cause is simple *rebuttal*, i.e. the fact that some items can not be believed simultaneously, because that would cause a contradiction.

1.2 Problems with Current Approaches

Despite the fact that there exist quite a number of implemented reason maintenance systems, there still remain some problems unsolved that we consider as elementary. They are independent of the approach chosen.

1. The first problem results from the fundamental design decision to separate the RMS component from the problem solver (as shown in fig. 1.1) which at first sight appears to be reasonable, since then the same RMS can be used for different problem solvers with minimal adaptation problems.

   However, some tasks a RMS is expected to perform are made difficult or even impossible by that separation. Because the RMS component has no clue about the semantical contents of the nodes it maintains it totally relies on the information transmitted by the problem solver. In order to detect a contradiction even in such a simple case, when two atomic propositions A and $\neg A$ are simultaneously believed in, it has to be informed about the existence of a contradiction, for the two propositions would be mapped to two different nodes in the dependency net, with no semantical relation between them. The net only stores the connections to those nodes that have been used in their derivation. The fact that they constitute a contradiction has to be *explicitly stated*. We call this the *interface problem*.

2. A second flaw concerns the semantics. Many systems have been implemented in an ad hoc fashion. In some cases semantics have afterwards been supplied, but the formalisms used for this task differ from system to system, thus making it impossible to *compare* various systems on that base.

3. These two problems taken together yield a third one: The separation between problem solver and RMS blocks the possibility to give a semantical characterization of the system as a whole, which is always an important problem. Take as an example the semantical characterization of de Kleer's ATMS: This states that the ATMS finds out *all* contexts in which an item can be derived. If e.g. a problem solver deals with the set of formulae $\{A, A \rightarrow C, B, B \rightarrow C\}$ and has currently found out that C can be derived using A and $A \rightarrow C$ (as well as modus ponens), and uses an

ATMS as its RMS component, the single context computed that entails C is $\{A, A \to C\}$, whereas the second possibility, namely $\{B, B \to C\}$ goes undetected. This is not the ATMS's fault. Concerning what has been transmitted to it this is certainly all it can detect. So it has in fact found out all the possibilities wrt. its knowledge. But this information, as correct as it may be, is rather useless. By far more interesting would be statements taking the combined components into consideration.

4. The fourth problem is the restriction of logics. In most systems justifications must be of a particular form, e.g. Horn formulae. Of course there have been generalizations defined, such as e.g. the CMS (clause management system) as a generalization of ATMS, but (for efficiency reasons) not the full power of the logic used by the problem solver is available for the RMS.

1.3 The Aim of this Thesis

In this thesis we define *labelled logics*. This will serve us as a tool to solve the problems listed in the previous section: It gives a genuine semantics to RMSs, using the same formalism for all the approaches. Moreover, we explicitly demonstrate the degrees of freedom where a system designer can decide what his system should be like, thus giving rise to manifold possibilities of combinations not yet found in existing systems. We do not first and foremost want to tell anything about how such a system could be implemented. In a later chapter (chapter 7) we demonstrate that it can be done by supplying a rather primitive generic method, albeit this may be inefficient.

The interface problem is solved, since labelled logics describe the whole system, including the problem solver as well as the (classical) RMS. The separation of the two systems is mirrored by the property of labelled formulae to be composed of two rather independent parts stemming from different logics. In practice the RMS part is completely taken over by the problem solver itself, whose procedures have to be altered only slightly in order to work on labelled formulae, the management of which incorporates the book-keeping of justifications and dependencies.

Finally, the problem of restricted logics in current approaches is completely disposed of.

1.4 Overview

The thesis consists of three main parts. The first (chapters 2–6) deals with Reason Maintenance and the problems presented above.

In chapter 2 we give a short introduction to Reason Maintenance, covering the main systems and the aspects of importance for our subsequent analysis. Also the notation and terminology needed for the more formal parts of the thesis are introduced here.

In chapter 3 we discuss the problems inherent in all approaches to Reason Maintenance in closer detail.

Only then does it seem appropriate to present our own approach, which is done in chapter 4. There, all the necessary formalism is introduced and the important theorems are proven. This chapter is in a sense the heart of the matter and the reader should have understood it thoroughly before proceeding.

Chapter 5 returns to the systems described in chapter 2 and gives their presentation within our framework. We also show how new systems can be described via their respective properties in our formalism at the end of that chapter.

We are then in a position to discuss the differences to other approaches to solve the semantics problem proposed in the past and we do this in chapter 6.

The other two parts are represented by chapters 7 and 8, which show possible extensions of the basic framework introduced in the first chapters.

Chapter 7 complements the theoretical analysis by giving guidelines for the design of practical systems. The necessary calculi are developed and it is shown what a system based on labelled logics would look like. It is here that we return to the abstract reasoner as described above.

In chapter 8 we digress from RMS and turn our attention to default reasoning. This is very close to RMS in several aspects, so it is no surprise that we are able to deal with that in a generalization of our framework.

We give a short summarizing overview of our results in chapter 10, also mentioning the directions that we consider interesting for further investigation.

It remains to note that parts of this thesis have already been published in (Fehrer, 1993) respective (Fehrer, 1994). The gist of the abstract reasoner construction is hinted at in (Fehrer, Hustadt, Jaeger, Nonnengart, Ohlbach, Schmidt, Weidenbach & Weydert, 1994).

Chapter 2

On the History of Reason Maintenance

This chapter will serve a twofold purpose. First we give a short survey of the historical development of reason maintenance. We shall, in particular, describe the systems of Doyle (Doyle, 1979) and de Kleer (de Kleer, 1986*a*) in detail, because they stand as representatives for the justification based resp. the assumption based class of approaches.

Besides, we use the chapter to introduce the main notions and definitions. We do not follow the respective authors' original notation, but introduce our own, as this will make it easier to compare the various systems with one another. The chapter is not intended to be a detailed introduction to Reason Maintenance. There exist textbooks that cover this topic (e.g. chapter 6, pp. 108–139 of (Pratt, 1994) has a nice presentation of TMS and ATMS as well as all the procedures; (Reinfrank, 1989) goes into greater depth).

2.1 The Early Beginnings

The seminal article of Doyle (Doyle, 1979) is usually seen as the starting point of reason maintenance. However, many of the ideas already appeared earlier in literature. One of the best examples is Stallman and Sussman's article on the diagnosis of switching circuits (Stallman & Sussman, 1977)[1]. They noticed that in problem solving it appears to be useful to learn from erroneous trials in order to avoid doing the same computations twice. So they propose systems which remember their reasoning

> "...threading the deduced facts with *justifications* which mention the antecedent facts used and the rule of inference applied."

[1]Diagnosis has always remained one of the most important areas of application for RMSs (besides prediction, i.e. model based reasoning); cf. e.g. (de Kleer & Williams, 1986*b*; de Kleer & Williams, 1987; Reiter, 1987).

> "The same justifications are employed by the system to determine the currently active database *context* for reasoning in hypothetical situations."

Nearly all of the key notions appearing in later articles (e.g. justifications, assumptions, context, well-founded support etc.) are already introduced in this article, but the most outstanding idea is the proposal to process *contradictions* via *dependency directed backtracking*, finding a *culprit* for the contradiction and trying to eliminate it.

The explicit conservation of dependencies in the derivation process has been used in many (expert) systems, such as e.g. TOPLE (McDermott, 1974) or MYCIN (Shortliffe, 1976). In the latter, however, the purpose for doing so was not the possibility of belief revision, but the generation of explanations.

2.2 Doyle's TMS

Doyle proposed the use of a special component for maintaining the dependencies between memory items, which he called *Truth Maintenance System* (TMS). This not very accurate nomenclature has been criticized, because what is managed is not truth but belief and reasons for belief. So even Doyle himself (in (Doyle, 1983)) supports the term

> "*'Reason maintenance systems'* (a less deceptive name than the original 'truth maintenance systems')."

Though "TMS" is still in use, more and more authors nowadays agree upon the more honest name "Reason Maintenance Systems". The term TMS (or sometimes JTMS, for justification-based TMS) is then reserved for Doyle's original system. We shall use it solely for this purpose.

The most important merit of Doyle's paper is the introduction of *nonmonotonic justifications*. This means that some of the recorded dependencies explicitly mark that some derivations depend on the *absence* of justifications for other items. As such a justification might be found and added later on, these derivations can be defeated. This is in contrast to the *monotonicity* property of classical logic, saying that the addition of further premises (axioms) can only increase, but never reduce the number of theorems.

2.2.1 Basic Definitions

A TMS consists of a set N of *nodes* and a set J of *justifications*. The nodes are atomic, whereas the justifications are triples, consisting of two sets of nodes (the so-called IN-SET and OUT-SET) and a single node, the *consequent*. All nodes in the IN-SET and the OUT-SET, taken together, are called the *antecedents*[2]. Intuitively a justification says: If all the nodes in the IN-SET and none of the nodes in the OUT-SET are believed, then the consequent should be believed.

[2] We sometimes speak of IN-antecedents and OUT-antecedents.

In order to denote what is believed and what is not, there is a *labelling function* $l : N \to \{\texttt{out}, \texttt{in}\}$, which assigns to each node a status, `out` meaning currently not believed and `in` meaning currently believed. It is worth mentioning that the disbelief in a node does not mean that its contrary (whatever that may be) is believed, i.e. we are talking about *justified belief*, and there may well exist situations where neither a particular proposition nor its contrary is justified.

Since the procedures which compute the status of the nodes proceed in an incremental manner and on their way deal with partial labellings (not every node has been assigned a status yet), in practice a third label, called `undet`, is introduced, saying the label of that node has not been computed up to now. Thus the labelling function is redefined to $l : N \to \{\texttt{out}, \texttt{in}, \texttt{undet}\}$. We shall not do this, because we do not take a closer look at how the algorithms proceed.

Let us put this more formally to fix the notation:

Definition 2.2.1 (Dependency Net)
A **dependency net** is a directed, bipartite graph DN $= (N, J, E)$ where

1. N is the set of **nodes**.
2. J is the set of **justifications**, $N \cap J = \emptyset$
3. inset, outset: $J \to 2^N$,
   conseq: $J \to N$.
4. $E \subseteq (N \times J) \cup (J \times N)$, the set of **edges**, such that $(x, y) \in E$, iff one of the following holds:
   - $x \in N$, $y \in J$, $x \in \mathsf{inset}(y) \cup \mathsf{outset}(y)$ or
   - $x \in J$, $y \in N$, $y = \mathsf{conseq}(x)$.

A justification j is sometimes denoted as a triple

$$< \mathsf{inset}(j), \mathsf{outset}(j), \mathsf{conseq}(j) > .$$

We often abbreviate dependency net by DN.

Definition 2.2.2 (Valid Justifications, Labelling Function)
The **labelling function** (or simply **labelling**) is given as $l : N \to \{\texttt{in}, \texttt{out}\}$.

A justification is said to be **valid**, iff all the nodes in its IN-SET are labelled `in` and all the nodes in its OUT-SET are labelled `out`.

Given arbitrary labels for all the nodes of a dependency net the validity of all justifications is uniquely determined by the previous definition. The intuition behind this is that the validity of an argument (which is what a justification stands for) depends solely on the validity of (or the status of belief in) the nodes involved.

The definition of the labelling function does not contain restrictions as to what label to assign to a node, so up to now the labels of nodes have been completely arbitrary. In order to interpret a dependency net as modelling real lines of argumentation, there are certainly some requirements that should be met by a labelling, e.g.

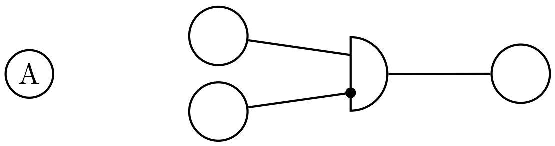
Definition 2.2.3 (Properties of Nodes and Labellings)
A labelling l of a dependency net is called **complete**, if the consequent of every valid justification is labelled `in`.

It is said to be **sound**, if for every node $n \in N$ which is labelled `in`, there exists a valid justification with n as consequent.

A justification is called a **premise justification**, iff its IN-SET as well as its OUT-SET is empty. A node which is the consequent of some premise justification is a **premise**. An **assumption justification** is a valid justification with nonempty OUT-SET. An **assumption** is a node with all its valid justifications being assumption justifications.

The requirement of completeness and soundness, taken together, is simply the translation of the demand that a node should be `in` exactly if it has a valid justification.

We do not present examples of dependency nets in the given notation, but instead use a familiar graphical representation, which is borrowed from switching circuits design[3].



a node called A

a justification with one node in the IN-SET and one node in the OUT-SET each

Nodes are represented as circles and justifications as semicircles ("bugs"). The lines drawn between them are interpreted as follows: There is only one line emanating from the curved side of a justification, leading to the consequent node, whereas arbitrarily many lines connect the straight side with members of IN-SET and OUT-SET. The latter are marked with a dot. In the switching circuit interpretation these are and-gates with negations in front of them. We sometimes give names to nodes in order to refer to them more easily. We then write the names into the respective circles. This is in contrast to the nodes' labels, which we always write outside the nodes.

We are only interested in complete and sound labellings. It is, however, not guaranteed that each dependency net possesses such a labelling, nor that there need be only one, as shown in figures 2.1 and 2.2.

In figure 2.1 there are two complete and sound labellings for the dependency net shown. One labels node A `in` and B `out`, and the other labelling is vice versa.

For the DN of figure 2.2 it is impossible to find a sound and complete labelling. If node A were labelled `out`, then the upper justification would be

[3]To my knowledge, this representation was first used for RMSs in (Goodwin, 1987).

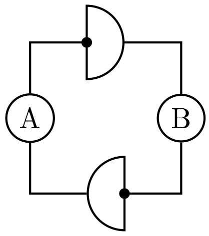


Figure 2.1: a dependency net with two sound and complete labellings

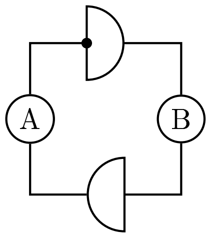


Figure 2.2: a dependency net with no sound and complete labelling

valid and thus B would have to be labelled `in`. This in turn would force the other justification to be valid also and thus make A `in`. A cannot be `in` either, because then there is no valid justification for B and therefore no reason for A to be `in`.

Of course the phenomenon of multiple or missing sound and complete labellings may be regarded desired in the two examples given. But what is certainly not satisfactory is the following: The dependency net in figure 2.3 possesses *two* sound and complete labellings, namely both nodes labelled `in` or alternatively both labelled `out`. One of those does not correspond to the usual intuition of justified belief, because the argumentation for both nodes marked `in` is clearly circular. This cannot be detected if one only uses local constraints like the soundness criterion.

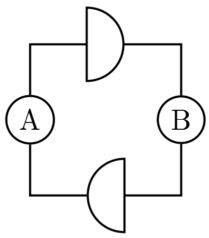


Figure 2.3: two sound and complete labellings due to circular argumentation

To rule out such pathological cases, labellings are restricted further to be *well-founded*. The well-foundedness condition is a stronger alternative for soundness and says that it is possible to *order* the nodes, so that every node being `in` is justified by justifications that depend only on nodes which are smaller in this ordering. This then precludes a circular argumentation. The ordering of the nodes is done via a *ranking* function, which is a mapping from nodes to the natural numbers.

Definition 2.2.4 (Well-Founded Labelling)
A labelling is said to be **well-founded**, if there exists a function $\mathsf{rank} : N \cup J \to \mathbb{N}$, such that for all nodes $n \in N$ the fact that n is labelled `in` implies that there is a valid justification j with n as a consequent and all the nodes x in $\mathsf{inset}(j)$ have a rank with $\mathsf{rank}(x) < \mathsf{rank}(n)$.

It is important to note that only the IN-SET is considered in the definition above. This stems from an asymmetry in the design, saying that every node can be assumed `out`, if there is no hint to the contrary, *without any well-founded justification*. This makes sense, since `out` does *not* mean that the contrary is believed.

Looking again at the examples, we find that the unintuitive labelling of figure 2.3 is now ruled out as not well-founded, whereas the DN of figure 2.1 still possesses two labellings.

2.2.2 CP-Justifications

Doyle introduced another kind of justifications, called CP-justifications, CP standing for conditional proof, besides the ordinary justifications we have considered so far, which he named SL-justifications (for support list). Doyle (1979):

> "A CP-justification is valid if the consequent node is *in* whenever (a) each node of the *in*-hypotheses is `in` and (b) each node of the *out*-hypotheses is `out`"

We shall not consider those any further, because — as many authors have pointed out — they can be translated to SL-justifications.

de Kleer (1986*c*):

> "Note that if, in the current database state, all the in-hypotheses are in and all the out hypotheses are out, this CP-justification is just the usual (i.e., SL) justification. Thus the CP-justification is simply a mechanism of recording or summarizing results which were determined in (usually) some other database state. In a sense, the ATMS which has no notion of current state, considers all justifications as CP, and translates each justification into all its corresponding SL-justifications in terms of the assumptions. As a consequence, the CP-justification is unnecessary for the ATMS."

A correct transformation is provided by Xianchang & Huowang (1991) (the transformation sketch in (Doyle, 1979) is shown to have some flaws in that article). We shall therefore not deal any further with CP-justifications in the sequel.

2.2.3 The Truth Maintenance Procedure

Doyle then presents two algorithms. The first one, the so-called *truth maintenance procedure* (TMP in the sequel), computes a well-founded, complete labelling for a given dependency net.

We do not present the TMP algorithm in closer detail. There exist many different implementations, that can be looked up in the literature. What is important is that it proceeds in a step by step fashion, moving from partial labellings (some nodes labelled `undet`) towards a complete one. The advantage of the algorithm is the *local* way of label propagation. Small changes in the dependency net (the only allowed changes being the addition of justifications) may in some cases concern the whole net, but most often the effects are only local. The algorithm exploits this fact and only relabels nodes in the vicinity of a node whose label has changed, if an `in` node has lost its last valid justification or an `out` node gets one.

The disadvantages, however, are manifold: If there are many possible labellings, one is chosen indeterministically. This is not really problematic. But, the arbitrary decision for one option may in some cases prevent finding an existing labelling at all.

The dependency net in fig. 2.4, e.g., possesses two sound and complete labellings. Suppose the TMP chooses the one shown in the figure.

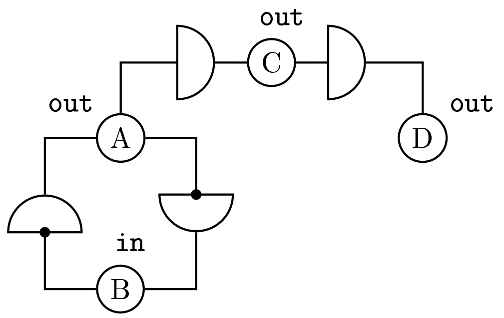


Figure 2.4: one of two possible labellings

Let us now introduce a further nonmonotonic justification from D to C. Local label propagation à la TMP relabels C to `in`, then D to `in` and then either goes into a cycle labelling C `out` again etc. or detects the loop and then terminates without result, depending on the implementation (fig. 2.5).

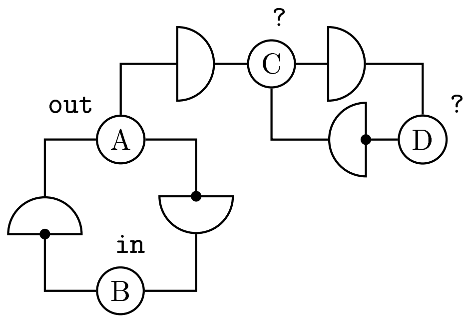


Figure 2.5: dead end for local labelling algorithms

However, had the TMS chosen the other one of the two possible labellings, namely the one given in fig. 2.6, the addition of the above justification would simply yield fig. 2.7.

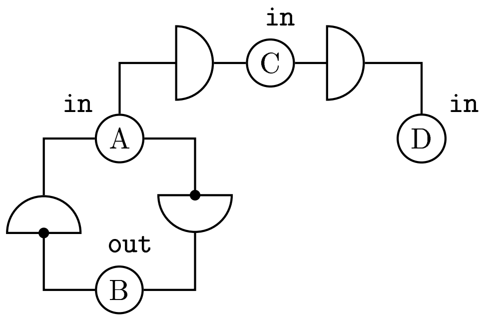


Figure 2.6: the second possible labelling

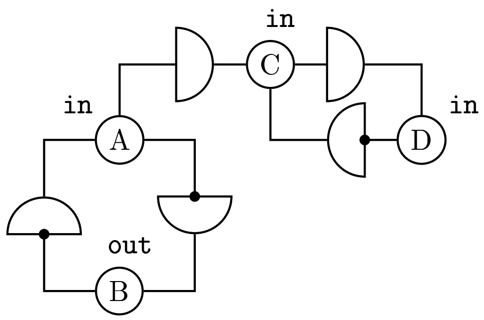


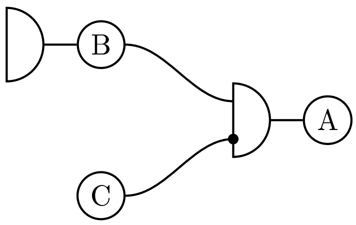
Figure 2.7: existing labelling

The example also shows, that the TMP algorithm faces difficulties when it comes across so-called *odd loops*. A loop is a cycle of edges in the dependency net. If one counts the dots on the path in the graphical representation (the number of times a member of an OUT-SET appears in the justification), one can classify loops into even and odd ones. Roughly speaking, even loops can be responsible for multiple candidate labellings and odd loops can prevent the existence of labellings. This is not completely true, but what can be said is, that a DN without odd loops is guaranteed to possess at least one well-founded complete labelling and a DN without any loops at all is guaranteed to possess exactly one. We have already encountered examples of such loops: Figure 2.1 is an even loop, and figure 2.2 represents an odd loop.

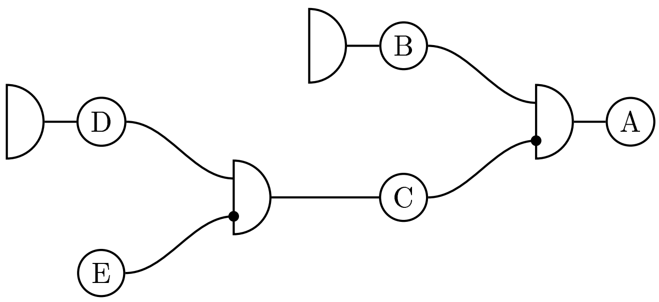
Some odd loops are detected by the TMP process, but some go undetected and cause the algorithm not to terminate, even in cases where a well-founded labelling exists.

The most serious disadvantage of TMS is the fact that the transition from one “belief state” to another is costly and clumsy. There are no means of enforcing belief or disbelief in an item other than by introducing new justifications. That means that in order to make a node retractable (i.e. specify belief in an item supposedly to be withdrawn later on) it must be based (beforehand) on a justification containing at least one node in the OUT-SET. Intuitively that means there is a dummy argument reserved for later defeat, which is not further specified. The former node can then be "withdrawn" by supplying the latter with a valid justification. A "redo" is only possible if this justification again is an assumption justification that mentions at least one node in its OUT-SET. Thus frequent "switches" blow up the dependency net.

To see this, assume there is a node A which represents a tentative assumption. Therefore it has to be provided with a possibility for later defeat, resulting in a net like



In order to retract A, node C must get a valid justification. If this should be possible to become "undone", it ought to get a defeasible justification with non-empty OUT-SET as well:

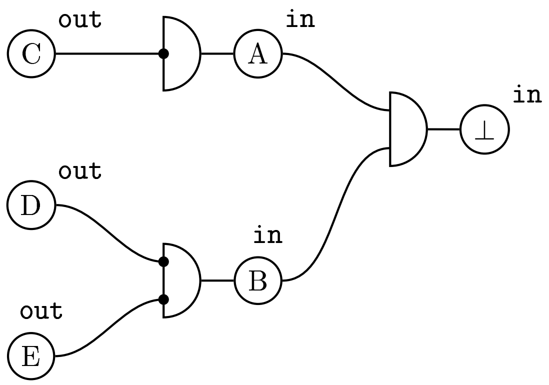


This procedure could be repeated arbitrarily often.

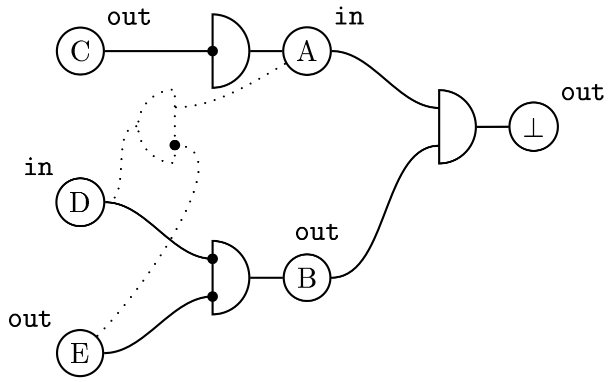
2.2.4 Dependency Directed Backtracking

There is a second procedure, which gained importance in literature: the *dependency directed backtracking* algorithm (DDB). Since the TMS has no clue about the semantical contents of its nodes, the problem solver is given the possibility of marking sets of nodes as incompatible with one another. This is done via the introduction of a distinguished *contradiction node*. We denote it by $\perp$. It bears the additional "semantics" that it may not be labelled `in`. If this should happen, the DDB procedure is invoked. Its task is to trace back the justifications, find a "*culprit*", and then render it harmless by destroying its justifications. This destruction is done by traversing in a backward direction through the justifications and supplying the first `out` labelled OUT-antecedent found with an artificially created valid justification. This justification mentions in its IN-SET the siblings of the culprit and in its OUT-SET other possible candidates of defeating OUT-antecedents[4]. An example may illustrate this:

Example 2.2.5



Here the contradiction node is labelled `in`, therefore DDB is invoked. There are two possibilities for a culprit, namely nodes A or B. If the decision is for B, again there are two alternative ways to proceed: Either node D or node E can be supplied with a valid justification. The following picture shows what the net would look like if D were chosen as the defeating node.



If there are no OUT-antecedent candidates, DDB cannot succeed. In most cases, however, there are numerous possibilities of culprits as well as of ways of defeating them. Hence DDB is highly nondeterministic. The justifications inserted into the DN by the DDB algorithm remain there and are not distinguishable from other justifications. Their presence thus influences the behaviour of the system in a way not always perceivable from the outside.

For the detection and management of odd loops algorithms are known that perform substantially better (see section 2.5). But the other flaws are inherent and gave rise to a completely different class of RMS, which we shall present now.

[4]We shall not exhibit the details here. The construction of the OUT-SET as mentioned is only a very cautious way of proceeding, for if any one of the nodes mentioned there becomes `out` resp. `in`, the newly constructed justification is simply unnecessary, since the node leading to the contradiction is then `out` anyway.

2.3 Multiple Contexts and de Kleer's ATMS

De Kleer's criticism of the TMS (de Kleer, 1984) can be summarized as follows:

1. the *single state problem*: Doyle's TMS produces one (global) state at a time. The system itself has no knowledge of "states". Thus it is not only very difficult to change states, but simply impossible to *compare* two of them. A state change can only be forced by introducing additional justifications (typically by declaring some nodes as contradictory). But as there is no way of eliminating justifications, these artificial justifications remain present in the system forever, and "toggling" a node between `in` and `out` results in lots of justifications accumulating in the dependency net, as shown in the previous section.

2. *contradiction avoidance*: The TMS does not admit any kind of contradiction, but immediately tries to return to a state where $\perp$ is labelled `out`. In connection with the single state property this means that the system has to decide for one possible consistent state and cannot follow alternative lines simultaneously, thus losing all consequences of nodes that were decided against. This is not adequate for every domain, as there are cases where some kind of *paraconsistency* is desired. If we have for example the (hypothetical) facts A and B, as well as the justifications $A \to C$, $B \to D$ and $A \wedge B \to E$, and then declare C and D to be contradictory, then the system will label either A or B `out`. Besides the desired consequence that of course E will also be `out`, we lose C resp. D in addition, depending on this decision. There is no way to see the "cases" "we can have C if A were `in`, we could have D if B were", corresponding to the two extensions of a default logic formulation of the example.

3. the *dominance of justifications*: Doyle's definition of an assumption depends on the structure of the dependency net. Adding a premise justification can change a node from an assumption to a premise. Often it seems more convenient to declare in advance what is to be viewed as assumption and what not.

   Besides, if asked why a node is `in`, the TMS could at best supply the complete tree of current support. Neither is it easy to see which assumptions this is based on, nor is it clear that there might not be an alternative way of justifying the node's status.

4. *single contexts only*: Pushing the latter argument even further, sometimes one would like to "switch" assumptions, e.g. in case analyses. The TMS may perhaps tell us that node n_2 is `in` whenever n_1 is. But it may well be the case that n_2 is completely independent of the state of n_1, and it is certainly interesting to find out about that.

These disadvantages led to the introduction of *assumption based truth maintenance* (ATMS, (de Kleer, 1986*a*)). Instead of the justifications, assumptions now play the primary rôle. In de Kleer's notation, assumptions are distinguished nodes representing basic items whose state of belief can be manipulated directly. All the other nodes' states are then derived by a procedure similar to the TMP.

To ease context switches, the labelling procedure does not compute a single state (`in` or `out`) for every node, but a list of all combinations of assumptions which force that node to be `in`. It is easy to see that this is a computationally complex task, however, once computed, the test for the state of a node *under arbitrary assumptions* consists of a simple lookup.

The label of a node in an ATMS represents a set of *environments*, i.e. a set of *sets of assumptions*. As the ATMS in its basic form does not support nonmonotonic justifications, addition of assumptions to an environment cannot cause a node to become `out`, if it was `in` before. So the label can be stored in a "minimal" form, containing only those environments that are not supersets of others already accounted for. This makes the label read like the distinguished disjunctive normal form (DDNF) of a propositional logic formula.

As in TMS, there is also the possibility of ruling out inconsistent sets of assumptions. This is again done by the introduction of a "contradiction node". Environments that make this node believed are called *nogoods*[5]. They are therefore eliminated from the labels of the nodes.

Aside from the obvious computational complexity of the ATMS procedure, the most prominent disadvantage in comparison to TMS is the lack of nonmonotonic justifications. Another flaw is not that easy to perceive: There is no such notion as an overall state of the system (what is believed and what is not). This can be seen as an advantage, because it permits digression from overall consistency, as already mentioned. But there may not even be a consistent state of global belief at all, which would then be difficult to detect[6].

Definition 2.3.1 (ATMS)
An ATMS is a quadruple (N, J, A, $\perp$), where N is a set of **nodes**, A a subset of N, the **assumptions**, and J is the set of **justifications**. $\perp$ is a distinguished element of $N \setminus A$, called the **contradiction node**. The nodes are atomic items, whereas a justification is a pair, consisting of a single node as **consequent** and a set of nodes as **antecedents**[7]. For the sake of convenience we use the functions inset and conseq the same way as with TMS (cf. definition 2.2.1) and denote a justification j as $< \mathsf{inset}(j), \mathsf{conseq}(j) >$.

An **environment** is a set of assumptions. A node n is said to **hold in an environment** e, if n is derivable[8] from $e \cup J$, provided the nodes are interpreted as propositional logic atoms and the justifications as Horn formulae with material implication.

A **label** is a set of environments. Labels are attached to nodes. A label l is called **consistent**, iff $\perp$ is not derivable from $l_i \cup J$ for all the environments

[5]The term nogood appears first in (Stallman & Sussman, 1977); Steele (1979) mentions nogood sets as well.

[6]In ordinary ATMS this argument does not apply, since the absence of nonmonotonic justifications guarantees that the empty environment (no assumption is `in`) is always consistent. This does not hold for nonmonotonic extensions of the ATMS.

[7]In de Kleer's notation there exists a third part, the so-called **informant**, but, as this is irrelevant for the considerations here, we shall not include it into our definition.

[8]De Kleer uses classical derivability here, and we shall simply follow him.

$l_i \in l$. It is **sound**, iff the node n, to which it belongs, is derivable from $l_i \cup J$ for all the environments $l_i \in l$. It is **minimal**, iff no two environments in l are subset resp. superset of one another. Finally, a label l is called **complete**, iff any consistent environment e with n derivable from $e \cup J$ is a superset of an environment in l.

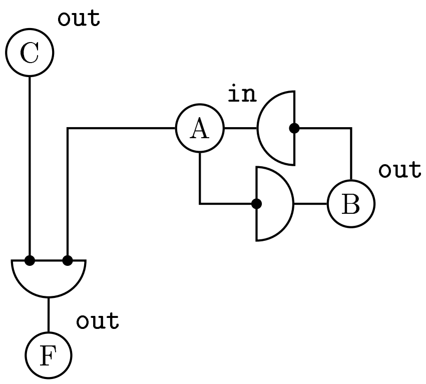
An environment e with $\perp$ derivable from $e \cup J$ is termed a **nogood**.

For every node the ATMS algorithm computes a label which is consistent, sound, complete and minimal.

2.4 Some Considerations on Dependency Directed Backtracking

The dependency directed backtracking procedure of the TMS has been criticized by various authors (including de Kleer) as not soundly motivated. DDB (as defined in (Stallman & Sussman, 1977) and used in (Doyle, 1979)) is problematic, because the system can not really decide *which* assumptions should be withdrawn[9]. So this is done in a nondeterministic way. Besides, as already mentioned, the "junk" justifications produced will stay within the dependency net forever. This can lead to wrong decisions, if the system chooses one particular labelling (of several possible) and then detects nogoods[10]. Returning to the still possible alternative may be impossible, because of the justifications introduced by the DDB process. The argumentation is similar to the one for the example in figure 2.2.3, but this time it is not the fault of the TMS procedure, for even globally informed search procedures cannot find the correct alternative, because the labelling is excluded by the justifications introduced by DDB.

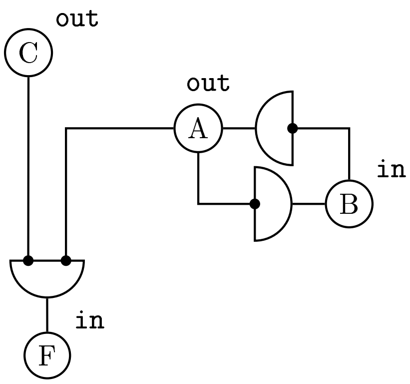
The following DN shall demonstrate this. It possesses two sound and complete labellings, namely



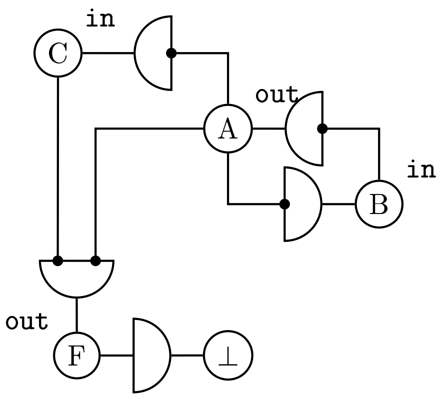
and

[9]In TMS terminology: which OUT-antecedents of assumptions should be given a valid justification.

[10]Of course this is ATMS terminology again. In TMS this corresponds to detecting that the contradiction node is labelled `in` by a labelling.



Now suppose the second labelling is chosen and node F is declared as contradictory, i.e. a justification of the contradiction node with F as sole antecedent is introduced. Because F is `in`, the DDB process is started. In search for a defeating node there are two choices: C or A could be given a valid justification. If C is chosen, the justification is constructed as

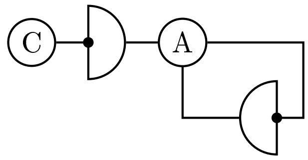


So the node C is forced to be `in`. This is completely unnecessary, for the alternative labelling (A `in`, B `out`) fulfills the same purpose without additional justifications. If the `in` state of C leads to contradictions later, it is impossible to return to that state without supplying A with an additional artificial justification. Even that would not suffice if the nogood strategy is not as cautious as in our example, where we included node A in the OUT-SET of the justification we created for C.

A thorough examination of this (TMS) problem is carried out by Elkan (1990). Elkan introduces the notion of a *correct* nogood strategy, by which he means that no unnecessary additional labellings are introduced. The algorithm of Petrie (1987), e.g., suffices this condition (under particular circumstances).[11]

Elkan gives a nogood strategy of his own, which he proves to be correct. He eliminates the special handling of contradiction nodes by the following construction: A contradiction node C is replaced by an ordinary node plus an additional node, call it A, and two justifications:

[11]Further thoughts about backtracking can be found in (Junker, 1990). The distinction between normal and backtracking justification is best discussed in (Brewka, 1990).



Elkan then shows that there is a one to one correspondence between labellings of the original dependency net that make the contradiction node `out` and labellings of the extended DN. This means that there are no "correct" labellings lost and there are none additionally introduced.

This is theoretically very nice, but fails to address part of why DDB was invented. Of course it is reasonable that, given a DN possesses several labellings and one is ruled out, one of the alternative labellings should be chosen instead of introducing a new one. But DDB is of particular importance for some DNs that do not have a well-founded labelling at all. In this case a correct nogood strategy must fail.

Nevertheless the Elkan idea is very useful, in that it describes what a DDB strategy should at least rule out and what it must not introduce *unnecessarily*. It is not that easy to describe when the necessity of an exception arises (one criterion might be the non-existence of a well-founded labelling) and what to do in such a case. Therefore in our modelling of TMS we shall from the start include a correct nogood strategy, and afterwards discuss an extension of how to do more DDB, but this extension suffers from the same flaws as does the Doyle approach.

Despite the criticism there are attempts to reintroduce backtracking into ATMS (de Kleer & Williams, 1986*a*). The cited article also contains a comparison of the two approaches. A more detailed elaboration of the differences is given in de Kleer (1986*c*), where besides Doyle's system (Doyle, 1979) also the systems described in (McAllester, 1980; McDermott, 1983; Martins, 1983; Martins & Shapiro, 1983) and others are taken into consideration.

2.5 Systems

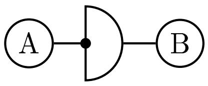
We have had a closer look at Doyle's TMS and de Kleer's ATMS in the preceding sections. Most other systems are reimplementations of either TMS or ATMS. In the TMS case there exist some real improvements concerning the detection of odd loops. There are cases in which Doyle's TMP does not terminate, even if a well-founded labelling exists. The WATSON system (Goodwin, 1987) performs better in some cases, but sometimes exits without result, too. Descriptions of improved labelling algorithms can be found in (Goodwin, 1982; Euzenat, 1992).

Another system to be mentioned here is RUP (McAllester, 1982; McAllester, 1980; McAllester, 1978), which is simpler, but more efficient than TMS. RUP[12] is clearly justification-based, but explicitly mentions assumptions. Therefore context switches are easier. Even temporary inconsistencies are possible. The problem of "unouting" a node is dealt with better than in TMS.

[12]Strictly speaking RUP itself is not a RMS, but it contains an RMS component.

In ATMS, there are numerous attempts to incorporate nonmonotonic justifications (de Kleer, 1986*b*; Dressler, 1989, among many others). This is not as easy as simply admitting nonmonotonic justifications, for this decision is accompanied by some fundamental changes in label representation.

For an example, consider the following dependency net, which should represent a nonmonotonic ATMS:



If A is an assumption, what should be the label of B? Clearly B holds in the empty environment. So the label of B should mention this environment, for it certainly fulfills all the requirements. But now we can not conclude, as in the basic ATMS case, that all supersets of this environment have the same property, for adding A destroys the derivability of B.

What is needed is a generalization of the set representation, very much the way it is done with the *restriction sets* used in the SNeBR system (Martins & Shapiro, 1983; Martins, 1983) described below. We shall talk about this issue in section 5.3.

The SNeBR[13] system of Martins and Shapiro is peculiar in some respects.

In contrast to all the other authors Martins and Shapiro do not favour the traditional separation between problem solver and RMS:

> "It would be desirable to put the responsibility of computing these dependencies on the system itself, so that as new beliefs are generated their dependency on old beliefs will *automatically* computed.
>
> This is a problem area that has been mostly ignored by researchers. The systems of Doyle, McDermott, and de Kleer do not address this issue at all: The inferences are made outside the system, which just passively records them."

So their abstract MBR (Multiple Belief Reasoner) model comprises the problem solving component, which is of a a natural deduction type, as well as the maintenance of dependency information.

For the latter they use a *relevance logic* they call SWM (for **S**hapiro, **W**and and **M**artins) in an assumption-based approach that allows for multiple contexts, without nonmonotonic justifications. The motivation for a relevant implication (Anderson & Belnap, 1975; Routley & Meyer, 1973; Dunn, 1986) is to avoid introduction of unnecessary (irrelevant) dependencies: If we e.g. have the two hypotheses "John is tall" and "John is fat", we could first conclude "John is tall and fat" using AND-introduction. Of course this depends on both hypotheses. By AND-elimination we could later obtain another "John is fat" from this, which we falsely assume to depend on *both* hypotheses again. If this

[13]SNeBR stands for SNePS with Belief Revision, where SNePS means Semantic Network Processing System.

is found to be part of a contradiction in the sequel, perhaps "John is tall" is blamed as a possible culprit, which is clearly nonsense.

The SWM logic is derived from the FR system of (Anderson & Belnap, 1975). It is described by supplying inference rules, and the authors explicitly ask the scientific public to provide a model theory for it. In chapter 5 we shall show that this can in fact be done using classical logics.

The implementation of nogood handling in SNeBR differs from ATMS. Instead of keeping a list of all known nogoods that is checked for necessary eliminations whenever a node's label is computed, Martins and Shapiro use what they call *restriction sets*. Nodes in SNeBR are labelled with an *origin set*, which is the environment in which the node was derived (note that there is only one single environment and not a set of them) and a restriction set[14]. The restriction set contains sets of assumptions that, when united with the origin set, yield a nogood. This representation is advantageous, because the restriction set of a newly derived node can be computed from the information contained in its parents without the need to consult a list of all nogoods detected, which then becomes obsolete. However, whenever a new nogood is detected, the restriction sets of all nodes have to be updated.

We shall look at this representation in closer detail in chapter 5, when we show how SNeBR can be represented in our framework. It is commented by de Kleer (1986*c*):

> "The idea of restriction sets presents an alternative implementation for nogoods because each union operation need only check the result against the restriction sets of its antecedents, not all nogoods."

but he adds in parentheses

> "This implementation of nogoods turned out to be inefficient for the ATMS"

a claim he does not explain further and which seems rather questionable.

The approach presented in (McDermott, 1983) is ATMS-like, but working with one single consistent context. There are labels in de Kleer's sense (sets of environments). Additionally, McDermott allows negated assumptions, therefore the label propagation algorithm is more complex (global constraint satisfaction instead of local propagation; but no special nogood database is needed).

De Kleer (1986*b*) suggests improvements for the ATMS, including the permission of disjunctions, nonmonotonic justifications, defaults (normal as well as non-normal) and arbitrary propositional formulae instead of atoms, but this is a purely algorithmic solution without any proofs. There is also a nonmonotonic ATMS version in (Dressler, 1989) which suffers from similar flaws.

The ATMS reimplementation in (Junker, 1987) directly follows de Kleer, incorporating in addition some parts from (de Kleer, 1986*b*) and (de Kleer, 1986*c*).

[14]There is also an *origin tag* that tells whether a node is a hypothesis, derived or "special". We do not consider this in detail.

Cayrol & Tayrac (1989) admit negation of assumptions. They introduce *generalized environments and nogoods*, an approach rather similar to our definitions in the respective chapters.

There is one system (Junker, 1989) which is truly nonmonotonic, yet computes all extensions and thus is not restricted to one single context. A specialty is the handling of exceptions to exceptions.

ATMS generalizations such as CMS are dealt with in the next chapter.

We have just described only the basic notions. Yet the details of different systems are mainly implementational issues and differences are motivated by the problems already mentioned or those presented in the following chapter. There is not much one can learn about the interesting concepts by looking in closer detail at single systems. The main categories suited to classify them are

1. whether they allow nonmonotonic justifications or not

2. whether they pursue the justification-based, single consistent state approach or are assumption-based.

Chapter 3

Shortcomings of Current Approaches

In spite of the fact that there have been quite a number of reason maintenance systems implemented up to now, some fundamental problems still remain unsolved. These are not only connected to just single systems, but are principally present, i.e. in all algorithms. The main problems are

1. The interface problem: the classic two component approach with strict separation between RMS and problem solver should be questioned.

2. Current systems are restricted to propositional logic, very often even to Horn clause logics[1].

3. Most of the systems have been given only an operational semantics, if any.

   - Even if there is a more abstract semantics, the formalisms used to describe various systems are too different to facilitate a comparison of approaches on this basis.
   - Frequently the given semantics skips the modelling of parts of the system; most often dependency directed backtrackingis omitted.

We discuss these and give a short preview on how we shall try to solve them in the sequel.

3.1 A Plea for a Combined System

Most authors and system designers share the belief that a strict separation of the problem solver and the reason maintenance component is a good idea, and traditionally this approach has been pursued ever since, starting with (Stallman

[1]Of course this has to be seen in connection with the two component approach. Although the logic of the RMS component may be propositional, this does not prevent the problem solver from using any arbitrary logic. This point, however, becomes problematic if one takes interactions (as e.g. the detection of inconsistencies by the RMS) into account.

& Sussman, 1977), and strongly supported in (de Kleer, 1984). Of course the possibility to modularize is appealing, for the inference mechanism could (at least theoretically) be exchanged without the need to change anything within the RMS. Thus the same RMS can be used independently of the type of problem solver attached to it, and especially independently of the logic it uses.

However, as we shall show, this advantage is traded against much of the potential power of an RMS. The modularization is based on the RMS treating every item it tackles as atomic and therefore deprived of any semantic content it may be given by the problem solver. This constitutes an *abstraction step*, because formulae of the problem solver's language are mapped to atomic formulae of some propositional language, giving up information that is potentially of value. For example it would be desirable if the RMS were capable of detecting *inconsistencies*. But given the abstraction mentioned, there is no way for the RMS to accomplish this. It can only manage contradictions if it is *explicitly told* about their occurrence. This places a heavy burden on the *interface* between the two components. Of course one could not expect the RMS to know of every contradiction inherently present, but it does not even know of every contradiction detected by the problem solver so far, but only of those transmitted to it.

Of course one can demand that the problem solver directly passes on every piece of information to the RMS. But even then the completeness of the RMS's contradiction detection depends on the problem solver's progress. This is what appears to be a little bit misguiding e.g. in the description of de Kleer's ATMS. There the computed labels claim to be "complete", and in fact they are, but only relative to what is known at the given moment. Any honest concept of completeness has to include both RMS and problem solver, so there is a limitation for modularization.

Even de Kleer (de Kleer, 1986*c*) states:

> "The advantages provided by the soundness, completeness and consistency of the ATMS can be inadvertently lost in the overall reasoning system."

therefore

> "...the ATMS is only one component of an overall reasoning system. This paper presents a set of concerns for interfacing with the ATMS, an interface protocol, ..."

He then talks about part of what we call the *interface problem* (what happens, if the problem solver transmits justifications that are either too general or too specific), but the proposed solution is described purely algorithmically (this also holds for all articles that try to implement this idea; a partial implementation is e.g. (Junker, 1987)), and the problem of a combined semantics for both components is not addressed. Proposals for how to improve the interface of de Kleer (1986*c*) can inter alia be found in (Dressler & Farquhar, 1990) (COntext driven COntrol).

The approach we want to propose in the next chapters overcomes the interface problem. We present a logic whose formulae consist of two separate parts, one representing the RMS and the other the particular problem solver's logic. We give a semantics for this logic, which incorporates the respective semantics of the components, thus giving the possibility to characterize the system as a whole and nevertheless preserve its modular character.

3.2 The Restriction of Logic

Independently of the nature of the logic used by the problem solver (if any), the RMS itself usually uses a very restricted logic. In one sense this is clearly an advantage: Since the nodes are considered as atomic, a propositional logic will do in any case for the RMS, no matter how sophisticated a logic the problem solver deals with. This is the reason for the decidability of the TMP and the ATMS label computation. But there are further restrictions imposed by the nature of justifications, which are again exploited in the design of algorithms. If translated into logic, it turns out that all the formulae resulting from an ATMS are Horn (or purely negative) clauses. Sometimes however one might like to have e.g. disjunctions within justifications. This was discovered very early, and Reiter & de Kleer (1987) came up with their CMS (clause management system). This is like the ATMS, but now the queries can be arbitrary (propositional) clauses instead of just atoms. The ATMS itself is a proper special case of CMS, with all the justifications being Horn clauses and nogoods purely negative clauses.

In this thesis we propose a framework that in principle admits arbitrary logics on the RMS side[2]. This has the theoretical advantage of comprising any RMS — given it is logically presentable at all. Of course one should restrict oneself as tightly as possible when designing a concrete implementation, for the performance or even the decidability of the respective algorithms certainly depends on this.

There are other attempts at using logics for providing general frameworks. Kakas & Mancarella (1990*b*) (resp. (Kakas & Mancarella, 1990*a*)) describe a nonmonotonic ATMS extension (including dependency directed backtracking) using a generalization of the stable models semantics from logic programming (Gelfond & Lifschitz, 1988) and autoepistemic logics (AEL) (Moore, 1985).

Elkan (1990) gives a logical characterization of TMS, again with the help of stable models and AEL.

The most general attempt at unifying the area was developed by McDermott. In (McDermott, 1991) he claims to be able to show how all the RMSs can be

> "unified into a single general RMS to see how the others are a special case of this one"

In order to do so McDermott replaces justifications by full propositional logic formulae, thus admitting contrapositives (cf. chapter 8 for a thorough discussion

[2]However, within this thesis we first deal only with classical propositional logic and later with first order logic.

of this topic) The resulting multidirectionality of justifications is called "logic based". This is certainly more general than e.g. TMS, but it is not shown what restriction could be imposed in order to characterize systems like TMS. The admittance of contrapositives certainly simplifies handling dependency directed backtracking, but yields results different from all JTMS approaches in cases like the Yale shooting example (again see chapter 8). An interesting point is that McDermott gives a semantics for odd loops.

In this approach multiple simultaneous assumption sets (as in the case of ATMS) are admitted. McDermott discusses the choice between the current context approach and the assumption based multiple context one and decides to use the second one without explaining how to model the first one. Nonmonotonicity is included via a multi-valued logic (valuations instead of simple interpretations).

The whole approach is completely procedural in nature. It does indeed contain most features one can think of (particularly multiple contexts as well as nonmonotonicity), but it can *not* serve as a general framework containing the "simpler" systems, as these features cannot be "switched off". McDermott's comment on this is

> "many of the algorithms used in my RMS would carry over to a single current-context framework (...) (of course, new algorithms would be required for handling nogoods, and for deleting justifications)"

There is no hint *how* the algorithms should be changed, hence the title "a general framework" is somewhat misleading.

3.3 A Genuine Model Theoretic Semantics

All implemented systems have in common the fact that they first have been characterized only operationally. That this constitutes a serious drawback, has been mentioned very early. E.g. Doyle (Doyle, 1983) writes:

> "To progress significantly beyond current reason maintenance systems, we must formulate their structure and intended behavior precisely enough to analyze computational complexities and tradeoffs independently of the current set of limited implementation proposals.
>
> There is little hope for improving on existing RMS implementations without clearer statements of their intended behaviors and better analyses of their performance. (...) These goals require mathematical formulations that clearly capture our intuitions ..."

It is true that some systems have been given a clear semantics afterwards, but often these semantics suffer from several drawbacks, as e.g.

- Some formalisms are developed just for the purpose of describing a particular RMS and therefore are very specialized and not generally known outside that area.

- Some do not fully characterize the systems in question but only part of it; in many cases this e.g. concerns DDB.

The most outstanding drawback is that the *semantics for different RMSs make use of different types of formalisms*, thus making it *nearly impossible to compare those systems to one another* (Katsuno & Satoh, 1991). So the main enterprise we shall undertake is to provide a *unifying framework* covering all of them. Thus it would in principle become possible to characterize the differences by telling what axioms hold in a particular approach. Besides, it appears that *new* systems can be described in a simple way just by enumerating the demands they should satisfy.

Chapter 4

The Labelled Approach

In this chapter we introduce the logic LL as a new approach to solving the problems presented in the preceding chapter. LL uses (and derives its name from) *labelled formulae*. Formulae of the logic used by the problem solver are labelled with information about the way they have been derived. Like many labelling methods in literature our mechanism has initially been developed independently, but later turned out to be a special case of Gabbay's Labelled Deductive Systems (Gabbay, 1991; Gabbay, 1994*b*). One significant difference is that the definition of an LDS has primarily been a purely syntactical matter, whereas show how to obtain a *semantics* for LL, provided the logic of the problem solver has one. In the meantime Gabbay showed how it is possible to combine semantics of different logics via a method called *fibering* (Gabbay, 1994*a*), and again our approach is a special case of that. We dedicate a later section (4.9)to this correspondence.

4.1 Labelling Formulae

Our main motivation in defining the *labelled logics* LL is to solve the problems of RMSs discussed above. However, as we shall see later (chapter 8), this approach also provides a means to deal with a broader class of problems. For the moment, let us suppose we are given a logic, henceforth called the *basic logic*, and we want to describe the dependencies between axiom sets and their respective sets of logical consequences. We pursue our goal by first attaching a *name* to every formula given as an axiom. We use Greek letters for these names, and we suppose them to be unique, i.e. no two formulae share the same name. An example might look like

Example 4.1.1

$$\begin{array}{l}\alpha{:}\text{RAIN}{\to}\text{WET}\\ \beta{:}\text{SPRINKLER}{\to}\text{WET}\\ \gamma{:}\neg\text{RAIN}{\to}\text{SPRINKLER}\\ \delta{:}\text{RAIN.}\end{array}$$

Since WET follows from RAIN and RAIN$\rightarrow$WET as well as from the collection of $\neg$RAIN$\rightarrow$SPRINKLER, RAIN$\rightarrow$WET, and SPRINKLER$\rightarrow$WET, we could denote this by something like $\{\{\alpha,\delta\},\{\alpha,\beta,\gamma\}\} \Rightarrow$WET. Instead we use the shorthand notation $\alpha\delta + \alpha\beta\gamma$:WET. It is easy to see that this in a sense corresponds to the *labels* in an ATMS, where the labels denote sets of *environments*, i.e. *sets of sets of assumptions*. The difference is that ATMS labels are assumed to meet some additional requirements, such as being complete (every minimal (wrt. set inclusion) assumption set that entails the consequent has to be mentioned), whereas up to now nothing prevents us from writing e.g. $\alpha\delta$:WET or $\alpha\beta\gamma\delta$:WET, both not being complete, the latter not even minimal. We call the respective property (to be defined in the sequel) of a label *maximality*[1].

The similarity of our label notation to algebraic or logic formulae is intended and gives a hint at the semantics we are aiming at. What we want to do is to treat the whole formulae (including their labels) as formulae of some kind of new logical language, the semantics of which is derived straightforwardly from the respective semantics of the logics of their label and formula parts. In all the definitions below we make use of different indices for notating the particular logic the symbols are to be interpreted in. We use BL for the **basic logic** (the logic the original formulae stem from), and PL for (classical) propositional logic, which will serve, at least for the moment, as the logic for the labels.

Concerning BL we want to make as few restrictions as possible, since we want to apply our approach to various types of logics. There are, however, some things we can assume without too much loss:

Nearly all of the theorems require that BL is *reflexive*. That means that a set of BL-formulae entails every formula contained in it. There are in fact very few logics that violate this assumption (we shall meet some in sections 4.5 and 4.7).

Sometimes we demand that BL be *monotonic*. Yet this is not as restrictive as it sounds, for the sole purpose of introducing nonmonotonicity into logic has been the desire to cope with default assumptions, which we are able to treat with monotonic basic logics as well (more to this in chapter 8).

This excepted, this section's theorems do not depend on any further properties of BL. Within the next section there are some theorems which assume particular properties of BL, the most prominent being the existence of a *falsum* (written $\perp$). This applies to all the parts dealing with consistent entailment.

Furthermore we suppose all sets of formulae to be *finite*.

Since we are mainly interested in semantics, we do not presuppose a particular calculus for BL. However, we assume that there *exists* a sound and complete calculus. Therefore we are not very strict concerning the use of words like "derivability" instead of "entailment". It sounds by far less complicated if we speak of multiple ways of derivation instead of stating that there are several

[1]The fact that we chose to call this criterion *maximality* may be a bit confusing, since the name resembles the minimality criterion for ATMS labels (cf. definition 2.3.1). However, it has nothing to do with it. Instead, our maximality correlates to de Kleer's completeness, as we shall soon see. We prefer this nomenclature, since later we shall approach maximal labels by constructing "bigger" and "bigger" ones, starting from very "small" labels, according to an ordering to be defined in definition 4.5.2.

subsets of a given set of formulae that all entail the formula in question, and it is intuitively clear what is meant. The same holds for the pair inconsistency/unsatisfiability.

To make the distinction clear between labels and formulae (since BL could of course also be PL) within labels we write conjunction by simple concatenation, denote negation by a bar over the label and write disjunction as "+". Also we omit most of the parentheses by assuming operator precedence of negation over conjunction over disjunction. So e.g. $(\alpha \wedge \neg\beta) \vee \gamma$ is written as $\alpha\overline{\beta} + \gamma$. We use $\top$ for the verum, $\perp$ for the falsum.

We almost always give labels in disjunctive normal form (DNF). The DNF of a formula is a logically equivalent formula which is either $\top$ or $\perp$ or consists of a disjunction of conjunctions of literals. There is a special disjunctive form called *distinguished disjunctive normal form* (DDNF). Usually this is defined as a DNF containing only the necessary prime implicants of a formula. Remember: an *implicant* of a formula φ is a conjunction of literals that implies φ. A *prime implicant* α of φ is an implicant, so that for any other implicant β of φ with $\alpha \rightarrow \beta$ follows $\beta \leftrightarrow \alpha$, that means leaving out any literal does destroy α's status of being an implicant. In a collection of prime implicants each one is *necessary*, if their disjunction is equivalent to φ, but none of them can be left out. E.g. $\alpha\beta$ and $\alpha\beta\delta$ are both implicants of $\alpha\beta + \gamma$, but only the former is a prime implicant. $\alpha\beta$, $\alpha\gamma$ and $\overline{\beta}\gamma$ are all prime implicants of $\alpha\beta + \alpha\gamma + \overline{\beta}\gamma$, but $\alpha\gamma$ is not necessary, because $\alpha\beta + \alpha\gamma + \overline{\beta}\gamma$ is equivalent to $\alpha\beta + \overline{\beta}\gamma$.

Thus defined, the DDNF representation of a formula is *not* unique. $\alpha\beta + \overline{\alpha}\overline{\beta} + \overline{\beta}\gamma$ and $\alpha\gamma + \overline{\alpha}\overline{\beta} + \alpha\beta$ e.g. are both DDNFs of the same formula. But in cases where the DDNF consists of prime implicants containing only positive literals, it is unique up to the ordering of implicants as well as literals within an implicant[2]. For more details see e.g. (Birkhoff & Bartee, 1970, an algebra textbook), or cf. articles like (Minicozzi & Reiter, 1972), (Slagle, Chang & Lee, 1969), (Reiter & de Kleer, 1987), the latter directly referring to our domain of discussion.

To circumvent the difficulties accompanying multiple DDNFs we digress from the usual definition and define our DDNF as

Definition 4.1.2 (DDNF)
The DDNF of a formula is the unique (up to ordering) disjunctive normal form that contains *all* prime implicants.

This is well-defined, since each formula has only a finite number of prime implicants[3]. Whenever we talk about DDNF in the sequel we mean this unique DDNF representation.

The computation of prime implicants is usually done by the so-called *consensus method* (Tison, 1967), but there are also incremental (Kean & Tsiknis, 1988) and other methods; e.g. Inoue (1990) uses an extension of linear logic.

[2]Strictly speaking this is not completely true. The formula $\top$ is an exception. It can be represented as $\alpha + \overline{\alpha}$ for an arbitrary atom α. We always choose $\top$ as *the* representation.

[3]Again consider last footnote for the only exception.

The reason why we choose to present labels in DDNF is that there is a direct correspondence between this form and sets of sets of axioms, as we soon shall see.

We need the following lemma:

Lemma 4.1.3
For any implicant β of a given formula φ there exists a prime implicant α of φ with $\beta \to \alpha$.

Proof:
Suppose β does not imply any prime implicant of φ. Then β can in particular be no prime implicant itself. Therefore there must be at least one implicant, say γ_1, of φ, which is implied by β, but not equivalent to it. Because of the structure of implicants, γ_1 must then consist of a set of literals which is a proper subset of the literals in β. If γ_1 itself is no prime implicant, there will be an implicant γ_2 of φ, which has still less literals and is also implied by β. Because of the decrease in the number of literals we finally come to a prime implicant γ_i of φ. But β implies γ_i. □

Definition 4.1.4 (formula and label function)
Given a labelled formula F, we refer to its label by **label**(F) and to the formula part by **formula**(F). If no confusion is possible, we simply speak of "formulae" when referring to labelled formulae. We further extend the functions label and formula to work on sets of formulae by defining

$$\mathsf{label}(\Phi) := \bigwedge_{F \in \Phi} \mathsf{label}(F)$$

and

$$\mathsf{formula}(\Phi) := \{\mathsf{formula}(F) \mid F \in \Phi\}.$$

Now we can make a first attempt at defining our logical consequence relation:

Definition 4.1.5 (Strict Labelled Logical Consequence)
Let Φ be a set of labelled formulae and $\alpha{:}F$ a single labelled formula. We say $\alpha{:}F$ **strictly follows** (logically) from Φ (written as $\Phi \models_{\mathrm{equivLL}} \alpha{:}F$), iff there exist subsets $\Psi_1, \ldots, \Psi_n$ of Φ ($n \geq 0$), with

- $\forall\, \Psi_i :\ \mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} F$ and
- $\models_{\mathrm{PL}} \alpha \leftrightarrow \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$.

Very often we shall meet *sets of sets* like the Ψ_i in the definition above. To increase clarity we talk about **collections** of sets in this context. Each Ψ_i is a set of assumptions, that — taken together — entail F. Different Ψ_i represent different ways of accomplishing this, corresponding to alternative ways of deriving F.

The definition is in several aspects more general than what we talked about before. Neither is there a demand that all the labels have to be different from one another, nor are they required to be atomic here. We have also said nothing about negative literals within the labels up to now, although they are not excluded by the definition. We shall not do this here, however, but rather permit arbitrary propositional formulae as labels, only making sure that the restricted case of unique, positive, atomic labels (see definition 4.1.8) is dealt with as expected.

Definition 4.1.6 (Basic, Semi-basic and Simple Sets, Assumptions)
A set of labelled formulae is called **basic**, iff every label is atomic and no label occurs more than once. It is called **semi-basic**, if in addition the occurrence of arbitrarily many formulae labelled $\top$ is allowed. The formulae not labelled $\top$ are called **assumptions**.

A set of labelled formulae is called **simple**, if in addition to the labels allowed in semi-basic sets conjunctions of atomic labels are allowed.

It is clear that we are mainly interested in basic sets. The rôle of assumptions and semi-basic sets will become apparent later, but we introduce them already here, because we can then prove our theorems in a more general form. As a guiding intuition let us say that the formulae labelled $\top$ can be used "for free", that means we do not track their use in labels.

It is immediately obvious that, given the set of formulae is basic, our definition of logical consequence corresponds to the intuitive set characterization, i.e. labels represent collections of sets of (unlabelled) formulae. So we always use the two views exchangeably.

In later chapters, when we are talking about calculi, we shall deal with operations that transform sets of formulae into other sets of formulae. There it will become apparent that some of the procedures do not preserve a set's status of being basic. For non-basic sets the correspondence between labels and collections of sets is not that obvious, and it will turn out that we would get difficulties in defining an appropriate notion of equivalence for sets[4], if we used the definition of strict labelled consequence.

We therefore introduce a second definition. For this, the correspondence to collections of sets is not immediately obvious (even for basic sets), but we shall show that for the cases of interest the new definition is equivalent to the former one.

Definition 4.1.7 (Logical Consequence on Labelled Formulae)
Let Φ be a set of labelled formulae and $\alpha{:}F$ a single labelled formula. We say $\alpha{:}F$ **follows** (logically) from Φ (written as $\Phi \models_{\mathrm{LL}} \alpha{:}F$), iff there exist subsets $\Psi_1, \ldots, \Psi_{\mathrm{n}}$ of Φ ($n \geq 0$), with

- $\forall\, \Psi_i : \ \mathsf{formula}(\Psi_i) \ \models_{\mathrm{BL}} F$ and
- $\models_{\mathrm{PL}} \alpha \rightarrow \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$.

We call the logic defined by the consequence relation $\models_{\mathrm{LL}}$ LL (for labelled logic).

[4]cf. the proof of theorem 4.3.10.

In order to prove the equivalence for the "interesting" cases, we first have to be more precise about what counts as interesting:

Definition 4.1.8 (Positive and Relevant Labels)
A label is called **positive** if its DDNF does not contain negative literals. It is called **relevant** wrt. a given semi-basic set Φ of labelled formulae, if each of its literals appears as a label of some formula in Φ. A labelled formula is said to be relevant resp. positive if its label is.

Now we can formulate the transformation between labels and collections of sets of formulae and thus justify our treating them as equivalent. For this task we use the so-called **characteristic function**.

Given a particular basic set Φ of labelled formulae, arbitrary positive, relevant labels can be looked upon as coding for sets of formulae, as each atomic label represents a unique formula. If transformed into DDNF, a label is either $\top$, $\bot$ or a disjunction of conjunctions of literals.

Definition 4.1.9 (Characteristic Function)
Let Φ be a semi-basic set of labelled formulae, and let $\mu(\alpha)$ for an atomic label α denote the set $\{F\} \cup \{G \mid \top{:}G \in \Phi\}$, where F is the unique formula F with $\alpha{:}F\in \Phi$. Then for positive, relevant α in DDNF the characteristic function χ is defined as

$$\chi(\alpha) = \begin{cases} \{\} & \text{if} \quad \alpha = \bot \\ \{\{\}\} & \text{if} \quad \alpha = \top \\ \bigcup\limits_{i=1}^{m} \{ \bigcup\limits_{j=1}^{n} \mu(a_{ij}) \} & \text{if} \quad \alpha = \sum\limits_{i=1}^{m} \prod\limits_{j=1}^{n} a_{ij} \end{cases}$$

(where the $\sum$ and $\prod$ stand for disjunction resp. conjunction). For positive, relevant α not in DDNF, we define $\chi(\alpha) = \chi(\text{DDNF}(\alpha))$.

Remark 4.1.10
The characteristic function always yields a collection of subsets *of* $\mathsf{formula}(\Phi)$.

Note that χ depends on Φ, so that we should correctly write e.g. $\chi_\Phi(\alpha)$. However, we omit this, since in most cases Φ remains fixed.

The inverse operation, getting a label from a collection of subsets of Φ, is best defined as a generalization of the known label function:

Definition 4.1.11
Let Θ be a collection of subsets of a set Φ of labelled formulae. Then define

$$\mathsf{label}(\Theta) = \bigvee_{\Psi \in \Phi} \mathsf{label}(\Psi)$$

For this definition one should bear in mind that $\bigvee \emptyset = \bot$ and $\bigwedge \emptyset = \top$.

Now it is easy to see that the two functions are inverses of each other in the following sense:

Lemma 4.1.12
Let Φ be a semi-basic set of labelled formulae. Then for relevant, positive α

$$\models_{\mathrm{PL}} \mathsf{label}(\chi(\alpha)) \leftrightarrow \alpha$$

holds, and if $\Theta = \Psi_1, \ldots, \Psi_n$ is a collection of subsets of Φ, no two of which are subsets of one another, we have

$$\chi(\mathsf{label}(\Theta)) = \bigcup_i \{\Psi_i \cup \{\top{:}G \in \Phi\}\}.$$

The proof is a trivial computation with elementary sets. For basic Φ the set of non-assumptions is empty, and thus we get

Corollary 4.1.13
If Φ is basic, then if Θ is a collection of subsets of Φ, no two of which are subsets of one another, we have

$$\chi(\mathsf{label}(\Theta)) = \Theta.$$

Theorem 4.1.14
Let Φ be a semi-basic set of labelled formulae, and assume the basic logic BL *is monotonic. Then $\models_{\mathrm{equivLL}}$ and $\models_{\mathrm{LL}}$ are equivalent for positive, relevant formulae, i.e. for all labelled formulae $\alpha{:}F$ with α positive and relevant wrt. Φ, we have $\Phi \models_{\mathrm{equivLL}} \alpha{:}F$ iff $\Phi \models_{\mathrm{LL}} \alpha{:}F$.*

Proof:

$\Rightarrow$: trivial (holds even without the restriction to positive, relevant labels).

$\Leftarrow$: Let $\Phi \models_{\mathrm{LL}} F$. Then there exists a collection of Ψ_i that satisfies the conditions of definition 4.1.7. Since α is relevant and positive, $\chi(\alpha)$ is defined and yields a collection $\Theta = \Psi'_1, \ldots, \Psi'_n$ of subsets of Φ. As $\alpha \to \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$ holds and the labels on both sides of the implication are positive, every Ψ'_j is a superset of some Ψ_i. Since BL is monotonic, F follows from all the Ψ'_j. Since $\alpha = \mathsf{label}(\chi(\alpha)) = \bigvee_{j=1}^{m} \mathsf{label}(\Psi'_j)$ as well, $\Phi \models_{\mathrm{equivLL}} F$. □

The following theorem shows the correspondence between $\models_{\mathrm{LL}}$ and the collections of sets view. It is crucial for the ATMS representation.

Theorem 4.1.15
Given a semi-basic set Φ of labelled formulae, let A *be the set of assumptions in Φ and* $\mathrm{J} = \Phi \setminus \mathrm{A}$. *Then for any positive, relevant label α we have $\Phi \models_{\mathrm{LL}} \alpha{:}F$ iff $\chi(\alpha) \models_{\mathrm{BL}} F$. Furthermore, $\Phi \models_{\mathrm{LL}} \alpha{:}F$ iff for any prime implicant α_i of α the union of the set S of* BL*-formulae defined by*

$$S = \{G \mid \beta{:}G \in \Phi \text{ and } \beta \text{ is an atom in } \alpha_i\}$$

with formula(J) BL-*entails F.*

Proof:
Directly from the definition of $\models_{\mathrm{LL}}$ and χ. □

The theorem tells us about the meaning of the non-assumptions in semi-basic sets. They are not tracked in the labels and can be added to every environment described by a label. In reasoning systems therefore non-assumptions are used for those items which are uninteresting, because

- they are supposed to be "hard facts", i.e. will never need to be retracted,
- they should not serve as explanations.

Furthermore we can see how positive relevant labels directly correspond to environments.

Computationally it is certainly better to have as few assumptions as necessary. But note that it is somewhat dangerous to declare too many things as non-assumptions: If the set of non-assumptions is BL-inconsistent, then we again find that everything follows, even in labelled logics (cf. section 4.3. Thus in our RMS characterizations the non-assumptions will be reserved for the axiomatic parts.

4.2 Labelled Deductive Systems

As defined, our labelled logic is just a special case of a *labelled deductive system* (LDS) (Gabbay, 1991; Gabbay, 1994*b*). Informally speaking, an LDS is a logical system working on formulae of some logic which are prefixed with additional information called labels. The original motivation was to code control information such as counting how often a particular formula has been used into a label, but the formalism admits labels of arbitrary form.

The inference rules of a respective calculus are now such that they operate on the labels as well as on the formulae itself. The operations on the labels therefore have to be defined. What one gets this way is a proof theoretical description of the combined system, i.e. a purely syntactic characterization.

When looking solely at the formula part of the labelled formulae, one notices that the constraints arising from taking the labels into account block the derivation of some formulae formerly derivable, thus resulting in a *change of logic*. This logic is by no means easy to be recognized in every case, though very interesting observations (such as mimicking linear logic by adding labels to classical logic) have been obtained even at that stage.

The whole matter becomes by far more interesting, if — as in our case — the labels themselves are taken from a logic and thus carry a semantics as well. Yet obtaining the combined semantics may be a sophisticated task. In our case we can give this semantics. We shall first do this in a way particularly fitted to the example at hand, but in section 4.9 we show that this can be obtained as a special case of a *fibred semantics* as proposed by Gabbay (Gabbay, 1994*b*; Gabbay, 1994*a*).

The rest of this chapter serves three purposes: First we prove some important properties of labelled logics, just to get a feeling about what we are dealing with (sections 4.3 and 4.4). Some of the theorems may seem to be a little unmotivated at this stage, but they will be used to prove the key theorems in later chapters in a rather concise and elegant way. Next we present further variations of labelled consequence (sections 4.5 to 4.7), which are needed to characterize different RMSs occurring in literature. Finally (sections 4.8 and 4.9) we present a true model theoretic semantics.

4.3 Some Useful Properties of LL

Suppose BL were classical predicate logic. If we start with an inconsistent set of axioms, we are able to prove anything, and usually this is not particularly welcomed. Instead, one often wants to keep inconsistency *local*. This is possible in our approach. It is not the case that *anything* follows from a database obtained from an axiom set that is inconsistent wrt. the basic logic. E.g. $\{\alpha$:A, β:¬A, γ:C$\}$ does imply $\alpha\beta\gamma$:D, but not β:D.

There are, however, sets of labelled formulae from which arbitrary labelled formulae follow. But this cannot happen with basic sets. $\top$:A e.g. is never entailed by a basic set Φ of formulae, if A is an atom not occurring in $\mathsf{formula}(\Phi)$, for this would mean that A is entailed by the empty set of BL-formulae. Note, however, that this does not hold for sets which are only semi-basic. Counterexamples are the sets $\{\top{:}A, \top{:}\neg A\}$ and $\{\top{:}\bot\}$.

Some further key properties independent of BL are

Proposition 4.3.1

1. *For every unlabelled formula F the labelled formula $\bot{:}F$ follows from an arbitrary set of labelled formulae.*

2. *If $\mathsf{formula}(\Phi)$ is inconsistent ($\mathsf{formula}(\Phi) \models_{\mathrm{BL}} \bot$) for a set Φ of labelled formulae, none of which is labelled with $\bot$[5], then there exists a label α different from $\bot$ with $\Phi \models_{\mathrm{LL}} \alpha{:}\bot$.*

3. *If F is a tautologous unlabelled formula (i.e. $\emptyset \models_{\mathrm{BL}} F$) then $\top{:}F$ follows from any set of labelled formulae.*

Proof:

The proof can be obtained easily, if one keeps in mind the correspondence of labels and collections of sets, particularly how the characteristic function maps the falsum and the verum. $\bot{:}F$ simply says: From all sets contained in the empty set (i.e. none) F can be derived (note: this does *not* mean: there is no subset of Φ from which F can be derived), which is clearly true for every Φ and F. For basic Φ $\top{:}F$ means F is entailed by the empty set, which is the definition of a tautology. The converse of 3 is not true even for semi-basic sets (it holds for basic ones). $\top{:}F$ only says F is entailed by the set of all non-assumptions in Φ. □

[5] We shall never make use of formulae labelled that way throughout the whole thesis.

There is one rather trivial lemma we shall often need in subsequent proofs. As an aside, note that this lemma does not hold for $\models_{\mathrm{equivLL}}$.

Lemma 4.3.2
If $\Phi \models_{\mathrm{LL}} \alpha{:}F$ and $\models_{\mathrm{PL}} \beta \rightarrow \alpha$, then $\Phi \models_{\mathrm{LL}} \beta{:}F$ as well.

Proof:
Immediately from the definition of $\models_{\mathrm{LL}}$. □

We have directly defined logical consequence without saying anything about validity up to now. In order to get a genuine model theory, however, we should be able to define that notion.

Definition 4.3.3 (Validity)
We say a labelled formula F is **valid** (written as $\models_{\mathrm{LL}} F$), iff $\emptyset \models_{\mathrm{LL}} F$.

We then immediately get the characterization

Lemma 4.3.4
A labelled formula $\alpha{:}F$ is valid iff $\models_{\mathrm{PL}} \alpha \leftrightarrow \bot$ or F is a BL*-tautology.*

Proof:
Let $\emptyset \models_{\mathrm{LL}} \alpha{:}F$. According to definition 4.1.7 there exist subsets $\Psi_i, i \geq 0$ of $\emptyset$ with particular properties. As there is only one subset of $\emptyset$, namely $\emptyset$ itself, the full definition reads as "either (n=1) $\emptyset \models_{\mathrm{BL}} F$ and $\models_{\mathrm{PL}} \alpha \rightarrow \top$ or (n=0) $\models_{\mathrm{PL}} \alpha \rightarrow \bot$", which yields one direction of the proof. The other one is elementary. □

Corollary 4.3.5
For arbitrary F the labelled formula $\bot{:}F$ is valid. If F is a BL-tautology, then $\alpha{:}F$ is valid for arbitrary α.

The next few properties directly characterize the consequence relation, as given in definition 4.1.7. In the sequel we make use of a generalization of $\models_{\mathrm{LL}}$ to sets.

Definition 4.3.6 (Consequence Relation on Sets)
By $\Phi \models_{\mathrm{LL}} \Psi$, Φ and Ψ being sets of labelled formulae, we mean that for all $\varphi \in \Psi$ we have $\Phi \models_{\mathrm{LL}} \varphi$[6]. For any set Φ of labelled formulae we define $\mathsf{Cons}(\Phi)$ as $\{\varphi \mid \Phi \models_{\mathrm{LL}} \varphi\}$.

One remarkable property of $\models_{\mathrm{LL}}$ is monotonicity[7]. Despite the fact that it can easily be proven, this is nevertheless noticeable, for we are attempting to characterize reason maintenance systems and it turns out that our consequence relation stays monotonic even in the presence of nonmonotonic justifications. But if we keep in mind the fact that we are not talking about the outcome

[6]Note that we interpret the set on the right hand side as a conjunction. This differs from some conventions in literature, such as e.g. (Gabbay, 1994*b*).

[7]Concerning the history of the treatment and phenomena of nonmonotonicity see (Bobrow, 1980), which is a whole issue on this theme.

of single queries (such as "does A hold?"), but characterize the "network" of detected dependencies of answers on sets of axioms, this is not that amazing any more, as this network is monotonically increasing even in a TMS[8]. Nevertheless, it is interesting to see that the following theorem is completely independent of the basic logic, i.e. BL may well be nonmonotonic.

Theorem 4.3.7 (Monotonicity of the Consequence Relation)
For any labelled formula φ and sets of labelled formulae Φ and Ψ holds:
If $\Phi \models_{\mathrm{LL}} \varphi$ and $\Phi \subseteq \Psi$, then $\Psi \models_{\mathrm{LL}} \varphi$.

Proof:
In definition 4.1.7 the existence of *subsets* Ψ_i is the crucial point. Of course these are also subsets of any superset of the original set of formulae. □

Not that obvious is the fact that one can add consequences to a set of formulae without increasing the set of logical consequences, i.e. that the consequence relation, when generalized to sets of labelled formulae, is transitive.

Theorem 4.3.8 (Transitivity of the Consequence Relation)
Let BL be monotonic (and reflexive). Then for any set Φ of labelled formulae $\mathsf{Cons}(\Phi) = \mathsf{Cons}(\mathsf{Cons}(\Phi))$.

Proof:

$\subseteq$: From the fact that $\models_{\mathrm{LL}}$ is reflexive (easy to see if BL is) we have $\mathsf{Cons}(\Phi) \models_{\mathrm{LL}} \mathsf{Cons}(\Phi)$. This means $\mathsf{Cons}(\Phi) \subseteq \mathsf{Cons}(\mathsf{Cons}(\Phi))$ by definition of the latter.

$\supseteq$: Let $\mathsf{Cons}(\Phi) \models_{\mathrm{LL}} \alpha{:}F$. Then there exist $\Psi_1, \ldots, \Psi_{\mathrm{n}} \in \mathsf{Cons}(\Phi)$ with $\forall\, \Psi_i : \mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} F$ and $\models_{\mathrm{PL}} \alpha \rightarrow \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$. If all the Ψ_i are subsets of Φ, we are done. So assume there is a Ψ_i containing a formula φ_j with $\varphi_j \notin \Phi$. Of course $\Phi \models_{\mathrm{LL}} \varphi_j$. Let $\varphi_j = \beta{:}G$. Then there exist $\Theta_1, \ldots, \Theta_{\mathrm{m}}$ such that for all Θ_i we have $\mathsf{formula}(\Theta_i) \models_{\mathrm{BL}} G$ and furthermore $\models_{\mathrm{PL}} \beta \rightarrow \bigvee_{i=1}^{n} \mathsf{label}(\Theta_i)$. Now replace Ψ_i by Ψ_{i1} to Ψ_{im} defined as $\Psi_{ik} = (\Psi_i \setminus \{\varphi_j\}) \cup \Theta_k$. For all Ψ_{ik} still $\mathsf{formula}(\Psi_{ik}) \models_{\mathrm{BL}} F$ holds because of the monotonicity of BL. In addition $\models_{\mathrm{PL}} \alpha \rightarrow \bigvee_{i=1}^{n} \mathsf{label}(\Psi_{ik})$. Still the sets contain only elements from $\mathsf{Cons}(\Phi)$. If they are not subsets of Φ, the same argument can be repeated and, as we have only finite sets, this will certainly terminate. Nevertheless the argumentation works for infinite sets also, because we can choose to replace all candidates in one single step. □

[8] The fact that nonmonotonicity does *not* apply to the justifications themselves, for their set is monotonically increasing, has been mentioned by de Kleer (1984, p. 79). Already Israel (1980) remarks, that inferences may be nonmonotonic, but *derivability* (entailment; and this is what we talk about) still remains monotonic.

This theorem states that *consequences* can freely be added to a set of labelled formulae without either enlarging or diminishing the number of formulae entailed. This property is also known under the name *cumulativity* (cf. section 4.7).

In the sequel we write $\varphi \equiv \psi$ to denote $\varphi \models_{\mathrm{LL}} \psi$ and $\psi \models_{\mathrm{LL}} \varphi$. φ and ψ can be single labelled formulae or sets of formulae. It is clear that $\equiv$ is an equivalence relation. Reflexivity and symmetry are obvious, if BL is reflexive, and transitivity follows from theorem 4.3.8.

The next theorems and propositions show us different equivalence preserving transformations on sets of labelled formulae.

First, if there is some equivalence relation defined on formulae of BL, we can freely substitute equivalent BL-formulae within labelled formulae, still preserving equivalence ($\equiv$) as defined above. The same holds for equivalence preserving transformations within the label part.

Proposition 4.3.9 (Equivalences in Label or Formula Part)
Given F and G are formulae of the basic logic, α and β are labels. Let $\leftrightarrow_{BL}$ be an equivalence relation on BL-formulae. Then the following equivalences hold:

1. *if $\models_{\mathrm{BL}} F \leftrightarrow G$, then $\alpha{:}F \equiv \alpha{:}G$ for arbitrary α.*
2. *if $\models_{\mathrm{PL}} \alpha \leftrightarrow \beta$, then $\alpha{:}F \equiv \beta{:}F$ for arbitrary F.*

The proof is trivial.

Theorem 4.3.10 (Contraction)

$$\{\alpha{:}F, \beta{:}F\} \equiv \alpha + \beta{:}F.$$

Proof:

$\Rightarrow$: trivial

$\Leftarrow$: this proof is elementary, too. But it is worth noting here that this direction does not hold for the naïve definition 4.1.5. In fact *this* is the reason for us to introduce definition 4.1.7 instead of 4.1.5. □

The next theorem does not hold for arbitrary basic logics, but only for those containing a classical "and" connective.

Theorem 4.3.11 (Splitting)
If there is a connective $\wedge$ in BL with the meaning $F \wedge G$ is satisfied iff both F and G are, then the following holds:

$$\alpha{:}F \wedge G \equiv \{\alpha{:}F, \alpha{:}G\}.$$

The proof is obvious.

It should be noted that transformations which preserve equivalence can nevertheless change a set's status of being basic or semi-basic. Reading contraction from left to right introduces non-atomic labels, whereas splitting yields multiple copies of atomic labels. But, as those transformations work in both directions, starting with a basic or semi-basic set maintains our correspondence to collections of sets, because we can simply apply this to the respective basic (semi-basic) set, which we can view as the canonical representation.

4.4 Towards Proof Procedures

Many deduction systems use particular normal forms for formulae. If BL is classical first-order predicate logic, one would e.g. like to use established refutation procedures. This often presupposes the existence of a *clause normal form* (CNF) and the availability of a deduction theorem, which is the topic of this section.

Theorems 4.3.9 to 4.3.11 provide us with a means to define a clause normal form for suitable basic logics: if BL itself possesses a clause normal form, this carries over to labelled formulae.

Definition 4.4.1 (CNF for Labelled Formulae)
Suppose there is a definition of a clause normal form for BL-formulae. A set Φ of labelled formulae is in *clause normal form*, iff

1. all formula parts are in CNF as defined for BL
2. all the labels are in DDNF
3. each BL-formula occurs at most once as a formula part of a labelled formula.

An algorithm to obtain this is:

Algorithm 4.4.2 (Transformation to Clause Normal Form)

1. transform all the formula parts to conjunctive normal form. This does not affect the labels at all.
2. if necessary, apply splitting to obtain clause normal form of the formula parts. With first order formulae this will be accompanied by variable renamings.
3. apply contraction to reduce the number of occurrences for each BL-formula to exactly one.
4. transform the labels to distinguished disjunctive normal form.

The soundness proof for this procedure is obvious. For 1 and 4 use proposition 4.3.9 (1) resp. (2). 2 is shown by theorem 4.3.11 and 3 by theorem 4.3.10.

Next we prove an important property: A (slightly altered) *deduction theorem* holds for labelled logic, if there is one for the basic logic.

Theorem 4.4.3 (Deduction Theorem)
Let Φ be a set of labelled formulae, α a label and $A \to B$ a formula of the basic logic. Furthermore let BL be monotonic and possess a deduction theorem[9]*. Then we have*

$$\Phi \models_{\mathrm{LL}} \alpha{:}A \to B \quad \Leftrightarrow \quad \Phi \cup \{\top{:}A\} \models_{\mathrm{LL}} \alpha{:}B.$$

Proof:

$\Rightarrow$: Given $\Psi_1, \ldots, \Psi_{\mathrm{n}}, \subseteq \Phi$, with $\forall\, \Psi_i$ $\mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} A \to B$ and $\models_{\mathrm{PL}} \alpha \to \bigvee \mathsf{label}(\Psi_i)$, construct a collection of $\Psi'_i = \Psi_i \cup \{\top{:}A\}$. Clearly $\mathsf{formula}(\Psi'_i) = \mathsf{formula}(\Psi) \cup \{A\}$, so because of the deduction theorem for BL we get $\forall\, \Psi'_i$ $\mathsf{formula}(\Psi'_i) \models_{\mathrm{BL}} B$. It is easy to see that $\models_{\mathrm{PL}} \alpha \to \bigvee \mathsf{label}(\Psi'_i)$, because the latter is identical to $\bigvee \mathsf{label}(\Psi_i)$.

$\Leftarrow$: Let $\Psi_1, \ldots, \Psi_{\mathrm{n}}, \subseteq \Phi \cup \{\top{:}A\}$, with $\forall\, \Psi_i$ $\mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} B$ and $\models_{\mathrm{PL}} \alpha \to \bigvee \mathsf{label}(\Psi_i)$. W.l.o.g. we can assume that $\top{:}A \in \Psi_i$ for all i (If not, since BL is monotonic, we could simply add it without destroying the fact that $\mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} B$). Using the deduction theorem for BL we get $\forall\, \Psi_i$ $\mathsf{formula}(\Psi_i) \setminus \{A\} \models_{\mathrm{BL}} A \to B$. $\models_{\mathrm{PL}}$ $\alpha \to \bigvee \mathsf{label}(\Psi_i)$ is unaffected by this operation. $\square$

This theorem is not formulated as generally as it could be: $\top$:A could be replaced by $\beta{:}A$ for arbitrary β, if $\alpha \to \beta$ holds. But we only need the special case given above.

Theorem 4.4.3 has an important corollary, which provides us with a justification for refutation proofs:

Corollary 4.4.4 (Refutation Proof)
If the basic logic contains material implication[10]*, we get*

$$\Phi \models_{\mathrm{LL}} \alpha{:}F \quad \Leftrightarrow \quad \Phi \cup \{\top{:}\neg F\} \models_{\mathrm{LL}} \alpha{:}\bot.$$

Proof:
By substituting $\neg F$ for A and $\bot$ for B in theorem 4.4.3 (keeping in mind the fact that $F \leftrightarrow (\neg F) \to \bot$). $\square$

4.5 Alternative Notions of Labelled Consequence

The definition of labelled consequence, as given in section 4.8, does not suffice as a description of what e.g. de Kleer's ATMS does. What is missing can best be described by the notion of *maximality*. Let us look at

[9]Let Ψ be a set of BL-formulae and $F \to G$ a BL-formula involving an implication operator "$\to$". Then $\Psi \models_{\mathrm{BL}} F \to G$ iff $\Psi \cup \{F\} \models_{\mathrm{BL}} G$.
[10]By this we mean that $A \to B$ is equivalent to $\neg A \vee B$.

Example 4.5.1

$$\begin{array}{l}\top\text{:RAIN}\rightarrow\text{WET}\\ \top\text{:SPRINKLER}\rightarrow\text{WET}\\ \alpha\text{:RAIN}\\ \beta\text{:SPRINKLER.}\end{array}$$

Suppose we give this example to an ATMS[11]. Then the ATMS computes labels for every atomic proposition that strongly resemble ours. For WET the ATMS e.g. computes something like $\alpha + \beta$:WET. This means, that $\alpha + \beta$:WET should be deducible in a formalism attempting to represent an ATMS. In labelled logic, as defined up to now, however, we can in addition deduce e.g. α:WET or even $\alpha\beta\gamma$:WET. What distinguishes $\alpha+\beta$ from these other labels is called maximality. Informally speaking, $\alpha+\beta$ represents all possible derivations of WET and mentions only those assumptions that are really necessary for the derivation. Thus, in a sense, $\alpha + \beta$ is a *better* label for WET, than e.g. α. We capture this by defining an ordering on labels. We say that a label α is *greater than* a label β (wrt. an unlabelled formula F and a set of labelled formulae Φ) iff $\Phi \models_{\mathrm{LL}} \alpha{:}F$ as well as $\Phi \models_{\mathrm{LL}} \beta{:}F$ and $\models_{\mathrm{PL}} \beta \rightarrow \alpha$ holds.

Let us compare our labels to ATMS labels more closely. As already mentioned (definition 2.3.1), an ATMS label satisfies four conditions

- it is *sound*, i.e. the node in question really is deducible in all environments mentioned,
- it is *minimal* in the sense that none of the environments is a subset of another one also contained in the label,
- it is *consistent*, i.e. none of the environments is a nogood,
- it is *complete*, i.e. every environment in which the node can be deduced is a superset of one of the environments in the label.

The first property clearly holds for our labels, too, because that is how they are defined. The second one is a syntactical matter. If we consider only positive, relevant labels this is the same as putting the label to DDNF.

The last two properties are *not* satisfied by our labels, not even for positive, relevant labels. We therefore need some additional means to express them.

First let us discuss the completeness criterion. The reason why we did not introduce this into the definition of labelled consequence is simple: A label represents ways to prove a formula. Usually we want to compute these incrementally. If the completeness criterion were incorporated into the consequence relation, intermediate results (we have not found all possible ways to prove the formula) are not sound in our logic. That means there is no way to generate lemmata on the way. In an ATMS this is exactly what happens. The ATMS

[11] Of course the notation in ATMS is quite different. For the moment the reader should just believe that this example can in fact be transformed to represent an ATMS problem.

computes the complete labels and only afterwards tells the result. The possibility to do this of course depends on the fact that the ATMS uses classical propositional logic for its labels and therefore this is decidable. We, however, do not want to restrict ourselves to this limited case, for we attempt not to treat BL-formulae as atomic and do not want to use only decidable basic logics. Besides that, a logic, where only complete labelled formulae are considered as sound, must necessarily be nonmonotonic, since adding further axioms may increase the number of different derivations for an item.

In the sequel we first introduce some properties one could demand of labels. Later we shall define various alternative consequence relations based on different combinations of these properties. All the properties refer to a given set Φ of labelled formulae in their definitions. Since we assume this Φ to remain fixed in our considerations, we shall most often not explicitly mention it.

First we define a (partial) ordering on labels, based on classical implication.

Definition 4.5.2 (Ordering on Labels)
We say a label α is **greater than** a label β, written $\alpha > \beta$, if $\models_{\mathrm{PL}} \beta \to \alpha$.[12]

Note the direction. Thus $\top$ is the "biggest" label, $\perp$ the smallest. In particular, for any α and β the relations $\alpha\beta < \alpha$ and $\alpha + \beta > \alpha$ hold.

Definition 4.5.3
We say α is **a label for** a BL-formula F (for given Φ), if $\Phi \models_{\mathrm{LL}} F$.

If we look at the labels for a given BL-formula, we can ask whether there is a "best one" concerning the ordering $>$ (intuitively it is clear why greater labels concerning $>$ are better, for these represent derivations showing more alternative ways or using less assumptions). Since $>$ is only a partial ordering, it is not clear that such a *maximal* label always exists. But indeed this is the case, as theorems 4.5.4 and 4.5.29 will show.

Theorem 4.5.4 (Existence and Uniqueness of Maximal Label)
Given an unlabelled formula F and a set of labelled formulae Φ there exists a label α (possibly $\perp$), such that $\Phi \models_{\mathrm{LL}} \alpha{:}F$ and α is maximal in the sense that for all labels β with $\Phi \models_{\mathrm{LL}} \beta{:}F$ also, $\beta \to \alpha$ holds. This α is unique up to logical equivalence.

Moreover, if Φ is simple, then α is relevant, and either $\perp$ or positive and thus unique up to the ordering of conjuncts and literals.

Proof:

existence: There is at least one label α, for which $\Phi \models_{\mathrm{LL}} \alpha{:}F$ holds, namely $\alpha = \perp$. There may well be many uncomparable labels, and in fact there are infinitely many of them, as there are that many atomic labels. We first show that only labels of the form $\bigvee \mathsf{label}(\Psi_i)$ with $\Psi_i \subseteq \Phi$ can be candidates for maximal labels. This is quite easy: Suppose $\Phi \models_{\mathrm{LL}} \alpha{:}F$. Then by definition there exist $\Psi_i \subseteq \Phi$ whose formula parts imply F and for which $\models_{\mathrm{PL}} \alpha \to \bigvee \mathsf{label}(\Psi_i)$ holds.

[12] We also use $<$ and "smaller" with the obvious meaning.

If (the DDNF of) α is not of the form given above, we can simply construct a greater label $\bigvee \mathsf{label}(\Psi_i)$, so α can not be maximal. So the maximal label (if it exists) must be of that form.

Next we notice that, given a finite Φ, there are only finitely many subsets and thus only finitely many α in the candidate set (up to ordering of both literals and conjuncts). Take all of them that are labels for F. It is easy to show that their disjunction is a label for F as well and is implied by every other label of F in the candidate set.

uniqueness: Suppose there are two such labels. As they are both maximal, both imply every other label of F, particularly they imply one another, i.e. they are equivalent.

relevance/positiveness: If Φ is simple, all the labels occurring in it are positive. We saw above that the maximal label is of the form $\bigvee \mathsf{label}(\Psi_i)$, which can directly be noticed to be relevant and positive or equal to $\perp$. □

Therefore we may define:

Definition 4.5.5 (Maximal Label)
For given Φ and F we call the label α guaranteed by theorem 4.5.4 the **maximal label** for F (wrt. Φ), written $\alpha = \mathsf{maxlabel}(F, \Phi)$, or, for known Φ, $\mathsf{maxlabel}(F)$.

We can now define an entailment relation incorporating the maximality criterion:

Definition 4.5.6 (Maximal LL-Entailment)
Let Φ be a set of labelled formulae. Φ **maximally LL-entails** $\alpha{:}F$, written $\Phi \models_{\mathrm{max(LL)}} \alpha{:}F$, iff

- $\Phi \models_{\mathrm{LL}} \alpha{:}F$ and
- for all labels β with $\Phi \models_{\mathrm{LL}} \beta{:}F \quad \beta \rightarrow \alpha$ holds.

Current systems like ATMS will produce states described by non-maximal labelled formulae only temporarily.

The following corollary to theorem 4.5.4 shows that maximally LL-entailed formulae are also strictly LL-entailed:

Corollary 4.5.7
Given a labelled set of formulae Φ and a BL-formula F,

$$\Phi \models_{\mathrm{equivLL}} \mathsf{maxlabel}(\Phi, F){:}F.$$

Proof:
In the proof above it is shown that the maximal label must be of a particular form. This formula can easily be detected as also *strictly* LL-entailed. □

We further have

Lemma 4.5.8
A label α which is a conjunction of atoms is a label for a formula F, iff there exists a prime implicant β of $\mathsf{maxlabel}(F)$ *with* $\alpha \to \beta$.

Proof:

$\Rightarrow$: If α is a label for F, then $\alpha \to \mathsf{maxlabel}(F)$ because of the definition of maxlabel. Because of the particular form of α it is an implicant of $\mathsf{maxlabel}(F)$. Lemma 4.1.3 then guarantees the existence of a prime implicant with the required property.

$\Leftarrow$: Since $\alpha \to \beta \to \mathsf{maxlabel}(F)$, α is also a label for F because of lemma 4.3.2. □

For modelling the handling of nogoods in the ATMS case, or as well contradiction nodes in JTMS, we must be able to rule out some labels which also are labels of the falsum (cf. de Kleer's consistency criterion). Labels of $\perp$ correspond to nogoods[13]. Actually the case is slightly more complicated, since it does not suffice that the whole label of a formula is a nogood, but also formulae whose label contains prime implicants that are nogoods, should be shut out. If e.g. $\alpha\beta + \gamma{:}F$ is derivable, but also $\alpha\beta{:}\perp$, but not $\gamma{:}\perp$, then we nevertheless do not want to have $\alpha\beta + \gamma{:}F$ as consistently entailed. This corresponds to the fact that in ATMS the labels should not *contain* nogoods. So we define

Definition 4.5.9 (Consistent Labels, ω-Free Labels)
A label α is said to be **consistent** (wrt. a given Φ), if for all prime implicants β of α we do not have $\Phi \models_{\mathrm{LL}} \beta{:}\perp$. It is called **$\omega$-free**, if for all prime implicants β of α we do not have $\models_{\mathrm{PL}} \beta \to \omega$.

The reason why we define ω-free labels becomes apparent if one takes into consideration that an entailment relation that should produce only consistent labels must be nonmonotonic and thus cannot be computed using local inference rules. Therefore we shall approximate this using inference relations that produce ω-free labels, for strictly increasing ω, until we reach the "best" label available for $\perp$.

When we want to model JTMS systems, we have got a finite set of nodes which we shall map to BL-atoms. We also have to describe *states*. A state is given by a well-founded, complete labelling (in the TMS sense, definitions 2.2.3 and 2.2.4). Such a labelling corresponds to a label (in our sense) which for all mentioned nodes entails the proposition represented by that node or its negation. We call this property *input completeness*.

The other definitions are needed because we want to be able to compute the labels of interest incrementally. So we can approach input completeness by computing Ω-satisfying labels for increasing Ω, starting with the empty set and then adding one BL-atom after the other until we reach the set of all BL-atoms occurring in the set under consideration.

[13] This is not completely true. If we define nogoods that way, we are more general than the usual nogood definition. Nogoods in the usual sense correspond to labels of $\perp$ which contain no disjunctions. So our nogoods are sets of nogoods in de Kleer's sense.

Definition 4.5.10 ((Strictly) *A*-Satisfying Labels)
For a particular BL-formula A a label α is said to be ***A*-satisfying** (wrt. a given Φ), if there exist $\Psi_1, \ldots, \Psi_n \subseteq \Phi$ such that

- $\forall \Psi_i : \mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} A$ or $\mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} \neg A$ (maybe both)
- $\models_{\mathrm{PL}} \alpha \rightarrow \bigvee_i \mathsf{label}(\Psi_i)$.

It is said to be **strictly *A*-satisfying**, if $\exists \Psi_1, \ldots, \Psi_n \subseteq \Phi$ such that

- $\forall \Psi_i : \mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} A$ or $\forall \Psi_i : \mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} \neg A$
- $\models_{\mathrm{PL}} \alpha \rightarrow \bigvee_i \mathsf{label}(\Psi_i)$.

It can immediately be seen that a label which is strictly A-satisfying, is also A-satisfying.

Remark 4.5.11
Given a set Φ of labelled formulae and a BL*-formula A, then α is strictly A-satisfying wrt. Φ iff $\models_{\mathrm{LL}} \alpha{:}A$ or $\models_{\mathrm{LL}} \alpha{:}\neg A$.*

The proof is obvious.

Lemma 4.5.12
Let α be A-satisfying (for some A and Φ). Then there are labels α_1 and α_2 such that $\models_{\mathrm{PL}} \alpha \rightarrow \alpha_1 + \alpha_2$ and both α_1 and α_2 are strictly A-satisfying.

Proof:
Looking at the definition of A-satisfying labels we divide the Ψ_i into two (not necessarily disjoint) subsets Ψ and $\overline{\Psi}$, namely those which BL-entail A and those that BL-entail $\neg A$. Define α_1 as $\bigvee_{\Psi_i \in \Psi} \mathsf{label}(\Psi_i)$ and α_2 as $\bigvee_{\Psi_i \in \overline{\Psi}} \mathsf{label}(\Psi_i)$. Then α_1 and α_2 are certainly strictly A-satisfying, furthermore $\models_{\mathrm{PL}} \alpha \rightarrow \alpha_1 + \alpha_2$ holds. $\Box$

We can prove an interesting corollary of this lemma, if we use the following trivial lemma:

Lemma 4.5.13
If α_1 and α_2 are positive labels, then every prime implicant of $\alpha_1 + \alpha_2$ is a prime implicant of either α_1 or α_2.

This lemma does not hold for arbitrary labels. As a counterexample consider $\alpha_1 = \beta\gamma\delta + \beta\overline{\gamma}\overline{\delta}$ and $\alpha_2 = \alpha\overline{\gamma}\delta + \beta\gamma\overline{\delta}$. $\alpha\beta$ is a prime implicant of $\alpha_1 + \alpha_2$, but neither of α_1 nor of α_2.

Now we have

Corollary 4.5.14
If α is positive and A-satisfying (for some A and Φ), and β is a prime implicant of α, then β is strictly A-satisfying.

Proof:
First find α_1 and α_2 as in lemma 4.5.12. Clearly $\models_{\mathrm{PL}} \beta \to \alpha_1 + \alpha_2$, because β implies α. Because of the particular form of β (conjunction of literals) and lemma 4.5.13 it must be the case that $\models_{\mathrm{PL}} \beta \to \alpha_1$ or $\models_{\mathrm{PL}} \beta \to \alpha_2$ (maybe both). □

Next we define some generalizations of the notions just introduced.

Definition 4.5.15 ((Strictly) Ω-Satisfying Labels)
Given a set Φ of labelled formulae, let Ω be a set of BL-formulae. A label α is called **(strictly) Ω-satisfying** if it is (strictly) A-satisfying for all the $A \in \Omega$. It is called **input complete**, if it is strictly Ω-satisfying for Ω being the set of all BL-atoms occurring in Φ.

Based on the properties of labels just introduced, we may define further entailment relations:

Definition 4.5.16 (Consistent LL-Entailment)
Let Φ be a set of labelled formulae and $\alpha{:}F$ a single labelled formula. We say $\alpha{:}F$ **consistently follows** from Φ (written as $\Phi \models_{\mathrm{consLL}} \alpha{:}F$), iff

- $\Phi \models_{\mathrm{LL}} \alpha{:}F$ and
- α is consistent. $\Phi \not\models_{\mathrm{LL}} \beta{:}\bot$.

Remember that a label α is consistent, if none of its prime implicants is a label of $\bot$. Because of lemma 4.5.8 this means that none of its prime implicant implies $\mathsf{maxlabel}(\bot)$.

This is a stronger restriction than simply forbidding α itself to be a label of $\bot$. If we e.g. have $\models_{\mathrm{LL}} \alpha{:}\bot$, then $\alpha + \beta$ is no label of $\bot$, but nevertheless inconsistent.

As already noted, maximal LL-entailment is nonmonotonic. The same holds for consistent LL-entailment, hence in particular the combination of both. In order to give a semantics to calculi that incrementally construct maximal, consistent labels, we introduce chains of consequence relations that are monotonic, and whose limit approaches maximal resp. consistent entailment.

Definition 4.5.17 (ω-Consequence)
Let Φ be a set of labelled formulae, $\alpha{:}F$ a single labelled formula, and ω a label. We say $\alpha{:}F$ **ω-follows** from Φ (written as $\Phi \models_{\omega\text{-LL}} \alpha{:}F$), iff

- $\Phi \models_{\mathrm{LL}} \alpha{:}F$ and
- α is ω-free.

This definition is needed to approach consistent LL-entailment. The rationale behind it is that, if we already know that ω is a nogood, we want to exclude all the environments containing it.

The ordering on labels introduced by classical implication carries over to an ordering on different ω-entailments.

Proposition 4.5.18
Given $\omega_2 > \omega_1$, then $\Phi \models_{\omega_2\text{-LL}} \alpha{:}F$ implies $\Phi \models_{\omega_1\text{-LL}} \alpha{:}F$. Furthermore for arbitrary ω $\Phi \models_{\omega\text{-LL}} \alpha{:}F$ implies $\Phi \models_{\text{LL}} \alpha{:}F$.

This means with increasing ω (more nogoods found) less can be deduced.

Proof:
The first condition in definition 4.5.17 remains unchanged, because it is independent of ω. So only the second one has to be proved. Suppose there is a prime implicant β of α with $\beta \to \omega_2$. Then of course $\beta \to \omega_1$ holds as well. □

$\models_{\perp\text{-LL}}$ and $\models_{\text{LL}}$ are *not* equivalent. The latter is weaker. The difference is, that $\models_{\text{LL}} \perp{:}F$ holds for arbitrary F, but, since this is also true for $F = \perp$, $\not\models_{\perp\text{-LL}} \perp{:}F$ for every F. This, however, is the only difference:

Theorem 4.5.19
If $\alpha \neq \perp$, then for arbitrary F $\models_{\perp\text{-LL}} \alpha{:}F$ iff $\models_{\text{LL}} \alpha{:}F$.

Proof:

- $\Rightarrow$ simple. there is one less condition in the definition.
- $\Leftarrow$ $\exists$ prime implicants β of α with $\alpha \to \perp$ is only possible, if $\alpha = \perp$, else it would not be a prime implicant. □

The crucial result is that the chain of ω-LL-entailment relations really approaches consistent LL-entailment for increasing ω[14]:

Theorem 4.5.20
$\models_{\text{consLL}}$ is equivalent to $\models_{\mathsf{maxlabel}(\perp)\text{-LL}}$.

Proof:

- $\Leftarrow$ Suppose there is a prime implicant β with $\models_{\text{LL}} \beta{:}\perp$. Because of the definition of $\mathsf{maxlabel}$ we immediately get $\beta \to \mathsf{maxlabel}(\perp)$.
- $\Rightarrow$ let β be a prime implicant with $\beta \to \mathsf{maxlabel}(\perp)$. We have $\models_{\text{LL}} \mathsf{maxlabel}(\perp){:}\perp$. Then all the more we get $\models_{\text{LL}} \beta{:}\perp$. □

Corollary 4.5.21
As a special case we get: If Φ is BL-consistent, then we have $\mathsf{maxlabel}(\perp) = \perp$.

Definition 4.5.22 ((Strict) Ω-Consequence)
Let Φ be a set of labelled formulae, $\alpha{:}F$ a single labelled formula, and Ω a set of BL-formulae. We say $\alpha{:}F$ **(strictly) Ω-follows** from Φ (written as $\Phi \models_{\Omega\text{-LL}} \alpha{:}F$ resp. $\Phi \models_{\text{str-}\Omega\text{-LL}} \alpha{:}F$), iff

- $\Phi \models_{\text{LL}} \alpha{:}F$ and
- α is (strictly) Ω-satisfying.

[14] Actually it may overshoot. $\models_{\top\text{-LL}}$ means nothing is entailed at all.

Definition 4.5.23 (Input-Consequence)
Let Φ be a set of labelled formulae, $\alpha{:}F$ a single labelled formula. We say $\alpha{:}F$ **input-follows** from Φ (written as $\Phi \models_{\mathrm{inputLL}} \alpha{:}F$), iff

- $\Phi \models_{\mathrm{LL}} \alpha{:}F$ and
- α is input-complete.

As a further entailment relation we need the combination of consistent and input consequence:

Definition 4.5.24 (Consistent Input-Consequence)
Let Φ be a set of labelled formulae, $\alpha{:}F$ a single labelled formula. We say $\alpha{:}F$ **consistently input-follows** from Φ (written as $\Phi \models_{\mathrm{cons,inputLL}} \alpha{:}F$), iff

- $\Phi \models_{\mathrm{LL}} \alpha{:}F$ and
- α is consistent and input-complete.

We can now extend the definition of maximal labels to all the entailment relations considered up to now. In order to do so we have to check whether under the additional requirements the existence and uniqueness of maximal labels is guaranteed as well. Concerning existence, this is not the case, but if there is a label satisfying the required properties at all, then there exists a unique maximal one. In order to prove this, we must first show that the respective properties of labels are conserved when labels are disjunctively combined. We only need to show this for positive labels, for we consider only simple sets of labelled formulae, for which maximal labels of any kind must be positive (cf. the proof of theorem 4.5.4).

Lemma 4.5.25
If α and β are both labels for a formula F, then $\alpha + \beta$ is also a label for F.

Proof:
There exist sets $\Psi_1, \ldots, \Psi_{\mathrm{n}}$ and $\Psi'_1, \ldots, \Psi'_{\mathrm{m}}$ whose formula parts all imply F and for which $\models_{\mathrm{PL}} \alpha \rightarrow \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$ and $\models_{\mathrm{PL}} \beta \rightarrow \bigvee_{i=1}^{m} \mathsf{label}(\Psi'_i)$ hold. Therefore we have also $\alpha + \beta \rightarrow \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i) \vee \bigvee_{i=1}^{m} \mathsf{label}(\Psi'_i)$. □

It should be noted that the trivial direction of contraction (theorem 4.3.10) is a special case of this.

Lemma 4.5.26
If α and β are two positive, ω-free labels, then $\alpha + \beta$ is also ω-free.

Proof:
Suppose there exists a prime implicant γ of $\alpha + \beta$ with $\gamma \rightarrow \omega$. Because of lemma 4.5.13 γ is also a prime implicant of α or β. So α and β can not both be ω-free. □

Corollary 4.5.27
If α and β are two positive, consistent labels, then $\alpha + \beta$ is also consistent.

Proof:
This can be obtained from lemma 4.5.26 and the fact that being $\mathsf{maxlabel}(\bot)$-free is identical to being consistent (theorem 4.5.20). □

Corollary 4.5.28
If α and β are two (strictly) Ω-satisfying labels, then $\alpha + \beta$ is also (strictly) Ω-satisfying. If α and β are two input complete labels, then $\alpha + \beta$ is also (strictly) input complete.

The proof is trivial.

Now we have

Theorem 4.5.29
Given an unlabelled formula F and a simple set of labelled formulae Φ, if there exists a label α, such that $\Phi \models_{\mathrm{LL}} \alpha{:}F$ and in addition fulfills any combination of the requirements

1. *α is consistent,*
2. *α is ω-free for a given ω,*
3. *α is input complete,*
4. *α is Ω-satisfying for a given Ω,*

then for this combination there also exists a label β that is maximal in the sense of theorem 4.5.4, which is unique up to logical equivalence. Furthermore it is relevant, positive, and therefore even unique up to the ordering of conjuncts and literals.

Proof:
The proof follows the same lines as that for theorem 4.5.4. The additional conditions cause no harm, because, as shown above, the disjunction of two consistent / ω-free / input-complete / Ω-satisfying labels is also consistent / ω-free / input-complete / Ω-satisfying.

The crucial difference is, however, that there may be no maximal label in case there is no label for F satisfying the side conditions at all. □

Based on this we can now proceed to define further entailment relations that include the maximality criterion.

Definition 4.5.30 (General Maximal Entailment)
Let Φ be a simple set of labelled formulae, and let $\models_{\mathrm{E}}$ be one of $\{\models_{\mathrm{LL}}, \models_{\mathrm{consLL}}, \models_{\mathrm{equivLL}}, \models_{\omega\text{-LL}}, \models_{\Omega\text{-LL}}, \models_{\Omega\text{-LL}}, \models_{\mathrm{inputLL}}, \models_{\mathrm{cons,inputLL}}\}$.
Φ maximally E-entails $\alpha{:}F$, written $\Phi \models_{\max(\mathrm{E})} \alpha{:}F$, iff

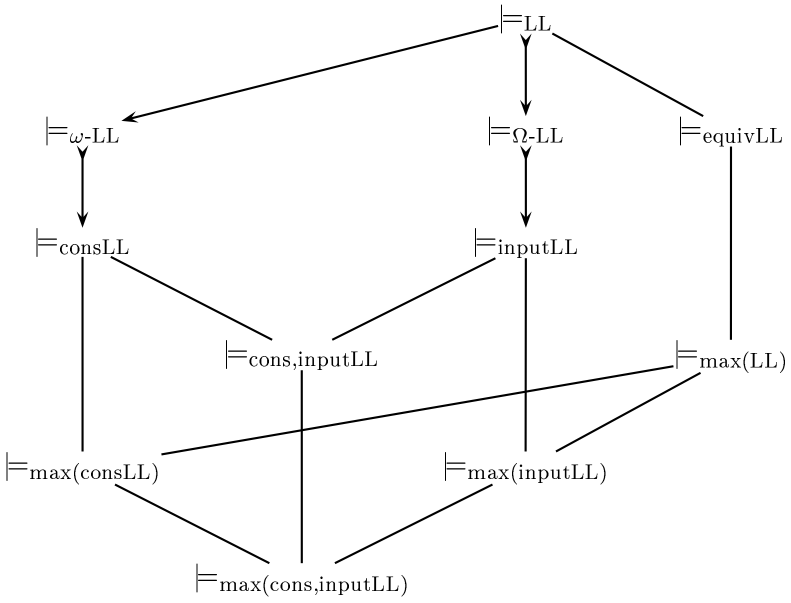
- $\Phi \models_{\mathrm{E}} \alpha{:}F$ and
- for all labels β with $\Phi \models_{\mathrm{E}} \beta{:}F \quad \beta \to \alpha$ holds.

Theorem 4.5.31
$\models_{\max(\mathrm{LL})}$ *and* $\models_{\max(\mathrm{equivLL})}$ *are the same.*

Proof:
Clearly $\Phi \models_{\mathrm{equivLL}} \alpha{:}F$ implies $\Phi \models_{\mathrm{LL}} \alpha{:}F$. So the maximal label according to $\models_{\mathrm{equivLL}}$ is at least a label concerning $\models_{\mathrm{LL}}$, too. But there can be no bigger one, because corollary 4.5.7 tells us that $\mathsf{maxlabel}(\Phi, F)$ is also a label according to $\models_{\mathrm{equivLL}}$. □

The interrelations between all the consequence relations defined so far can be summarized as follows:

Theorem 4.5.32
There are the following relationships between the entailment relations:



where lines (read downward) indicate proper set inclusion, the arrowheads symbolize approximation.

Proof:
Most parts have already been proved. The relations between $\models_{\mathrm{LL}}$, $\models_{\omega\mathrm{LL}}$ and $\models_{\mathrm{consLL}}$ are explained by theorems 4.5.19, 4.5.18 and 4.5.20. All the inclusions

can simply be read off the respective definitions, since they each represent additional conditions. We give counterexamples to show that they are all proper:

$\{\alpha+\beta{:}F\} \models_{\mathrm{LL}} \alpha{:}F$, but not $\{\alpha+\beta{:}F\} \models_{\mathrm{equivLL}} \alpha{:}F$.
$\{\alpha{:}F, \beta{:}F\} \models_{\mathrm{equivLL}} \alpha{:}F$, but not $\{\alpha{:}F, \beta{:}F\} \models_{\mathrm{max(LL)}} \alpha{:}F$.
The latter example can also be used for the pairs $\models_{\mathrm{consLL}}$ vs. $\models_{\mathrm{max(consLL)}}$, $\models_{\mathrm{inputLL}}$ vs. $\models_{\mathrm{max(inputLL)}}$ and $\models_{\mathrm{cons,inputLL}}$ vs. $\models_{\mathrm{max(cons,inputLL)}}$.
$\{\top{:}\bot\} \models_{\mathrm{max(LL)}} \top{:}F$, but not $\{\top{:}\bot\} \models_{\mathrm{max(consLL)}} \top{:}F$.

The proof that the Ω-LL relations form a chain with

$$\Omega_2 \supset \Omega_2 \quad \text{implies} \quad \Phi \models_{\Omega_2\text{-LL}} \varphi \quad \Rightarrow \quad \Phi \models_{\Omega_1\text{-LL}} \varphi,$$

$\models_{\mathrm{LL}} = \models_{\emptyset\text{-LL}}$ and $\Phi \models_{\mathrm{inputLL}} = \Phi \models_{\{\text{atom } A \mid A \text{ occurs in } \Phi\}\text{-LL}}$ is trivial. □

4.6 Some Computation Rules

We want to incrementally compute labels satisfying certain conditions, hence we must be able to tell which properties of labels remain unchanged by which operations. In particular we want to be able to compute maximal consistent labels that are input complete for a given set Φ.

We can not give a procedure for the general case, but if we restrict ourselves to the case that Φ is simple, we can proceed as follows:

First we can compute the maximal label (according to $\models_{\mathrm{LL}}$) for every atomic formula A. Then we obtain an A-satisfying label $\alpha+\beta$ from the maximal labels α for A and β for $\neg A$. The next lemma shows that this label is the *maximal* A-satisfying label.

Lemma 4.6.1
If α is the maximal label for A and β the maximal label for $\neg A$, then $\alpha+\beta$ is the maximal A-satisfying label.

Proof:
Suppose we have a γ that is A-satisfying. Because of lemma 4.5.12 this is decomposable into $\gamma_1+\gamma_2$, that are both *strictly* A-satisfying and hence a label for A or $\neg A$. Since α and β are maximal, both γ_1 and γ_2 must imply either α or β. Therefore we have $\gamma_1+\gamma \to \alpha+\beta$. Because this holds for all γ we obtain that $\alpha+\beta$ is maximal. □

Now the maximal A-satisfying labels for all atoms are conjunctively combined, yielding the maximal input complete label.

Lemma 4.6.2
If α is A-satisfying and β is B-satisfying, then $\alpha\beta$ is $\{A, B\}$-satisfying. Furthermore, if α is maximally A-satisfying and β is maximally B-satisfying, then $\alpha\beta$ is also maximally $\{A, B\}$-satisfying.

Proof:
The first part follows trivially from lemma 4.3.2. For the second part suppose that a γ is $\{A, B\}$-satisfying. This means in particular that γ is A-satisfying,

which in turn yields that $\gamma \to \alpha$ because of the maximality of α. The same holds for β and thus we have $\gamma \to \alpha\beta$. $\square$

The maximal consistent, input complete label is finally obtained from eliminating from the DDNF of the maximal input complete label all those prime implicants that are labels of $\perp$, i.e. imply a prime implicant of $\mathsf{maxlabel}(\perp)$.

The resulting label is of course consistent (because of the construction) and input complete (because it is a smaller label and thus implies everything the bigger one implied). But the maximality remains to be shown:

Lemma 4.6.3
If α is the maximal input complete label, then deleting in its DDNF all the prime implicants that are labels of $\perp$ yields the maximal consistent, input complete label.

Proof:
Let α be maximally input complete. Its DDNF can be written as $\alpha = \alpha_1 + \cdots + \alpha_n + \alpha_{n+1} + \cdots + \alpha_m$, where $\alpha_{n+1} + \cdots + \alpha_m$ are the prime implicants that are labels of $\perp$. Define α' as $\alpha' = \alpha_1 + \cdots + \alpha_n$. Clearly α' is consistent and input complete. Now suppose there is a γ that is suspected to be bigger than α' and fulfills the same properties. In order to be a candidate for this, γ must be positive. Since γ is input complete, we have $\gamma \to \alpha$ because of the maximality of α. As both α and γ are positive, each prime implicant of γ must imply a prime implicant of α. But since γ is also consistent, it can not be the case that any one of its prime implicants implies one of $\alpha_{n+1}, \ldots, \alpha_m$. But then γ must imply α'. $\square$

These lemmata give us a procedure for incrementally computing maximal consistent and input complete labels.

4.7 Properties of the Different Entailment Relations

In this section we analyze some important properties of the various entailment relations, as defined in the previous section.

Obviously not all of them are monotonic. In the literature on nonmonotonic entailment there are minimal requirements on an entailment relation in order to classify it as well-behaved in some sense. According to Gabbay (1985)[15] there are e.g.

monotonicity

$$\frac{\Phi \models \varphi}{\Phi \cup \Psi \models \varphi},$$

transitivity (cut)

$$\frac{\Phi \models \varphi; \Phi \cup \{\varphi\} \models \psi}{\Phi \models \psi},$$

[15]Other catalogs can be found in (Gärdenfors, 1988; Gärdenfors, 1990; Kraus, Lehmann & Magidor, 1990; Makinson, 1989).

restricted monotonicity

$$\frac{\Phi \models \varphi; \Phi \models \psi}{\Phi \cup \{\varphi\} \models \psi},$$

cumulativity (cut plus restricted monotonicity)

If $\Phi \models \varphi$, then $\Phi \models \psi$ and $\Phi \cup \{\varphi\} \models \psi$ are equivalent.

reflexivity

$$\forall \varphi \in \Phi\ \Phi \models \varphi.$$

As it turns out, *all* of our entailment relations with the exception of $\models_{\Omega\text{-LL}}$ are cumulative (and therefore transitivity and restricted monotonicity hold as well). This is independent of the monotonicity of BL.

Theorem 4.7.1
The properties of the entailment relations are as listed in table 4.1. The statements about reflexivity and transitivity presuppose that BL is reflexive.

	monotonicity	cumulativity	reflexivity
$\models_{\text{LL}}$	+	+	+
$\models_{\text{equivLL}}$	+	+	+
$\models_{\text{consLL}}$	−	+	−
$\models_{\omega\text{-LL}}$	+	+	−
$\models_{\max(\text{LL})}$	−	+	−
$\models_{\max(\text{consLL})}$	−	+	−
$\models_{\Omega\text{-LL}}$	+	+	−
$\models_{\text{inputLL}}$	−	−	−
$\models_{\max(\text{inputLL})}$	−	−	−

Table 4.1: Properties of the different entailment relations

Proof:

monotonicity: For $\models_{\text{LL}}$ this has been proved in theorem 4.3.7. For $\models_{\text{equivLL}}$ the proof runs the same. The additional constraints in $\models_{\omega\text{-LL}}$ and $\models_{\Omega\text{-LL}}$ present no difficulties either. For $\models_{\text{consLL}}$ the following may serve as a counterexample:

$\{\alpha{:}A\} \models_{\text{consLL}} \alpha{:}A$, but the addition of $\top{:}\overline{A}$ will block this. The same is a counterexample for $\models_{\max(\text{LL})}$, if instead $\beta{:}A$ is added. Again the same set with addition of $\beta{:}B$ refutes the monotonicity of $\models_{\text{inputLL}}$. The combinations are trivial then.

reflexivity: Here we have to assume that BL is reflexive. The case is trivial for $\models_{\text{LL}}$ and $\models_{\text{equivLL}}$ again. Counterexamples can be obtained by trying to derive $\alpha{:}A$ from the following sets:

cons	$\{\alpha{:}A, \top{:}\overline{A}\}$
max	$\{\alpha{:}A, \beta{:}A\}$
input	$\{\alpha{:}A, \beta{:}B\}$
$\alpha\beta$-LL	$\{\alpha{:}A\}$
$\{B\}$-LL	$\{\alpha{:}A\}$

The combinations are again easy.

cut: For $\models_{\text{LL}}$ this has been proved in theorem 4.3.8. For $\models_{\text{equivLL}}$ the proof is essentially the same, as are the proofs for $\models_{\omega\text{-LL}}$ and $\models_{\Omega\text{-LL}}$.

Theorem 4.3.8 tells us that, if $\Phi \models_{\text{LL}} \varphi$, then the addition of φ to Φ does not give us more theorems. This means that anything entailed by $\Psi \cup \{\varphi\}$ is already entailed by Φ itself. Therefore particularly the maximal label of an (unlabelled) formula stays the same. This also holds for $\perp$, i.e. $\mathsf{maxlabel}(\perp, \Phi) = \mathsf{maxlabel}(\perp, \Phi \cup \{\varphi\})$, if $\Phi \models_{\text{LL}} \varphi$. From these considerations the transitivity of the "max" and "cons" forms of LL-entailment follow.

That $\models_{\text{inputLL}}$ is not transitive, can be shown by the following counterexample:
$\{\alpha{:}A\} \models_{\text{inputLL}} \alpha{:}B \vee \neg B$, but also $\{\alpha{:}A\} \models_{\text{inputLL}} \alpha{:}A$, whereas $\{\alpha{:}A, \alpha{:}B \vee \neg B\} \models_{\text{inputLL}} \alpha{:}A$ does not hold.

restricted monotonicity: Those relations which are monotonic are of course also restrictedly monotonic. Thus it remains to give proofs for the rest.

As all of the relations are defined in terms of $\models_{\text{LL}}$, we can again exploit theorem 4.3.8 for this purpose. The additional conditions within the definitions do not present any difficulties. This time even $\models_{\text{inputLL}}$ is included. □

4.8 Semantics of Labelled Logics

Earlier we have claimed to be able to present a genuine model theory for the proposed labelled logic. The semantics described up to now does not on the first view satisfy this demand, since even though we reduced the definitions to consequence relations which are known to possess a model theory ($\models_{\text{PL}}$) or may be expected to have one ($\models_{\text{BL}}$), the process of combining these component theories is not as clear as to immediately make obvious how *interpretations* in labelled logics should look like.

In the following paragraphs we shall fill this gap. We define a logical consequence relation in the usual way in terms of interpretations and prove equivalence to our first definition. The details of such a proof depend on the logic used for the labels. In our case this is PL, but for digressions from that (and we shall present an alternative label logic in chapter 8) will need different model theories than those given in this chapter.

In the sequel we shall therefore stick to our former approach, because it emphasizes the compositional character and allows us to switch from one label logic to another more easily, since *this* definition stays the same *syntactically* for all label logics and only the meaning of the connectives within the label part changes depending on the particular logic.

It should be noted that fibering gives a possibility to directly obtain the combined semantics, as we shall see in section 4.9.

Definition 4.8.1 (Interpretations, Models)
Let $\Im_{PL}$ denote a propositional logic interpretation and $\Im_{BL}$ an interpretation of the basic logic in question[16]. Then a labelled logic **interpretation** $\Im_{LL}$ is a pair $(\Im_{PL}, \Im_{BL})$. $\Im_{LL}$ is said to be a **model** for a labelled formula φ (written $\Im_{LL} \models_{\mathrm{LL}} \varphi$) iff $\Im_{PL} \models_{\mathrm{PL}} \mathsf{label}(\varphi)$ implies $\Im_{BL} \models_{\mathrm{BL}} \mathsf{formula}(\varphi)$. $\Im_{LL}$ is a model for a set Φ of labelled formulae, if it is a model of all the formulae in Φ. A formula $\alpha{:}F$ is a logical consequence of a set Φ of formulae, iff

$$\forall \Im_{LL} : \Im_{LL} \models_{\mathrm{LL}} \Phi \;\Rightarrow\; \Im_{LL} \models_{\mathrm{LL}} \alpha{:}F.$$

It is important to see that this definition only applies to the special case where the label logic is classical propositional logic. Propositional interpretations $\Im_{PL}$ are referred to, and the interpretation of implication as material implication is "coded" into the definition via the formulation "$\ldots \Im_{PL} \models_{\mathrm{PL}}$ $\mathsf{label}(\varphi)$ implies $\Im_{BL} \models_{\mathrm{BL}} \mathsf{formula}(\varphi)$ ...", which should be read $\Im_{PL} \not\models_{\mathrm{PL}}$ $\mathsf{label}(\varphi)$ or $\Im_{BL} \models_{\mathrm{BL}} \mathsf{formula}(\varphi)$ ...". Thus definition 4.1.7 can be generalized more easily. Nevertheless the following important theorem holds:

Theorem 4.8.2
The two definitions of $\models_{\mathrm{LL}}$ (4.1.7 and 4.8.1) are equivalent.

Proof:

$\Rightarrow$: Let $\exists \Psi_1, \ldots, \Psi_{\mathrm{n}} \subseteq \Phi, n \geq 0$, with $\forall \Psi_i : \mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} F$ and $\models_{\mathrm{PL}} \alpha \rightarrow \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$ hold.

Now take an $\Im_{LL} = (\Im_{PL}, \Im_{BL})$ which is a model of Φ. We show that $\Im_{LL}$ is also a model of $\alpha{:}F$. If α is (PL-)unsatisfiable ($= \perp$), we are done, for then $\Im_{PL} \not\models_{\mathrm{PL}} \alpha$ and so $\Im_{LL} \models_{\mathrm{LL}} \alpha{:}F$. Otherwise the collection of Ψ_i contains at least one element (i.e. $n > 0$), or else $\bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$ would be $\perp$, which contradicts the fact it is implied by a satisfiable α.

Since $\Im_{LL} \models_{\mathrm{LL}} \Phi$, also $\Im_{LL} \models_{\mathrm{LL}} \Psi_i$ holds for all the Ψ_i. This means

$$\forall \Psi_i \forall \varphi_j \in \Psi_i : \Im_{PL} \not\models_{\mathrm{PL}} \mathsf{label}(\varphi_j) \text{ or } \Im_{BL} \models_{\mathrm{BL}} \mathsf{formula}(\varphi_j).$$

Suppose $\Im_{PL} \models_{\mathrm{PL}} \alpha$. Then $\Im_{PL} \models_{\mathrm{PL}} \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$. This means — as $n > 0$ — there is a Ψ_i with $\Im_{PL} \models_{\mathrm{PL}} \mathsf{label}(\Psi_i)$. This is equivalent to $\exists \Psi_i \forall \varphi_j \in \Psi_i \; \Im_{PL} \models_{\mathrm{PL}} \mathsf{label}(\varphi_j)$. As $\Im_{LL}$ satisfies

[16] Of course we do not claim to supply a combined model theory if the basic logic itself does not possess one. We assume the semantics of BL is given in the form: $A \models_{\mathrm{BL}} B$ iff in all (here may be added attributes like e.g. preferred) models of A B holds.

Φ this means $\exists \Psi_i \forall \varphi_j \in \Psi_i\ \Im_{BL} \models_{\mathrm{BL}} \mathsf{formula}(\varphi_j)$, which means $\Im_{BL} \models_{\mathrm{BL}} F$ because of our assumption that $\forall \Psi_i : \mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} F$.

$\Leftarrow$: Define the collection $\{\Psi_1, \ldots, \Psi_n\}$ as $\{\Psi_i \subseteq \Phi \mid \mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} F\}$. Now let $\forall \Im_{LL} : \Im_{LL} \models_{\mathrm{LL}} \Phi \Rightarrow \Im_{LL} \models_{\mathrm{LL}} \alpha{:}F$ hold. To show: $\models_{\mathrm{PL}} \alpha \rightarrow \bigvee_i \mathsf{label}(\Psi_i)$. Let $\Im_{PL} \models_{\mathrm{PL}} \alpha$ for an arbitrary $\Im_{PL}$.

Define $\Psi = \{\varphi \in \Phi \mid \Im_{PL} \models_{\mathrm{PL}} \mathsf{label}(\varphi)\}$ and $\overline{\Psi} = \Phi \setminus \Psi$. Suppose $\mathsf{formula}(\Psi) \not\models_{\mathrm{BL}} F$. Then there exists a basic logic interpretation $\Im_{BL}$ with $\Im_{BL} \models_{\mathrm{BL}} \mathsf{formula}(\Psi)$, and $\Im_{BL} \not\models_{\mathrm{BL}} F$. But this cannot be true, because then $\Im_{LL} = (\Im_{PL}, \Im_{BL}) \models_{\mathrm{LL}} \Phi$ (to see this, note that $\Phi = \Psi \cup \overline{\Psi}$, $\forall \varphi \in \Psi : \Im_{BL} \models_{\mathrm{BL}} \mathsf{formula}(\varphi)$ and $\forall \varphi \in \overline{\Psi} : \Im_{PL} \not\models_{\mathrm{PL}} \mathsf{label}(\varphi)$), but $\Im_{LL} \not\models_{\mathrm{LL}} \alpha{:}F$, which contradicts our assumption.

Therefore $\mathsf{formula}(\Psi) \models_{\mathrm{BL}} F$, which means $\Psi \in \{\Psi_i\}$. As $\forall \varphi_j \in \Psi\ \Im_{PL} \models_{\mathrm{PL}} \mathsf{label}(\varphi_j)$ (definition of Ψ), $\Im_{PL} \models_{\mathrm{PL}} \Psi$, and — as $\Psi \in \{\Psi_i\}$ — $\Im_{PL} \models_{\mathrm{PL}} \bigvee_i \mathsf{label}(\Psi_i)$. Since $\Im_{PL}$ was chosen arbitrarily, we are done. □

The case with all the other entailment relations is not that easy. Here it is not possible to give a characterization in the form e.g. $\Phi \models_{\mathrm{consLL}} \alpha{:}F$ iff every model of Φ is also a model of $\alpha{:}F$.

What is worse, we cannot get rid of restrictions of a rather syntactical flavour, namely the use of the notion of a prime implicant, which is not characterized in terms of entailment only, but depends on a particular syntactical structure (conjunction of literals).

In order to come to a model theoretic definition of consistent LL-entailment we could define notions like a subset of the models of a particular formula being "prime", if it can be expressed as intersection of the sets of models of atomic formulae or their complement wrt. the set of all interpretations. We shall not do this, because it is futile, because the notion of a formula being atomic is syntactical in nature, too. Therefore we leave the regress to syntax in the definition below.

Definition 4.8.3 (α-models)
Given a set Φ of labelled formulae and a label α, we say a model $(\Im_{PL}, \Im_{BL})$ of Φ is said to be an **α-model**, if $\Im_{PL} \models_{\mathrm{PL}} \alpha$.

Theorem 4.8.4
Let interpretations be defined as above. Then $\Phi \models_{\mathrm{consLL}} \alpha{:}F$ iff

1. *every model of Φ is also a model of $\alpha{:}F$, and*
2. *for every prime implicant β of α there exists a β-model of Φ.*

Proof:
As concerns the first property, the proof of theorem 4.8.2 can be used. The second property exactly corresponds to the additional restriction in definition 4.5.16 and can be handled by simply inserting the respective definitions. □

We hope that this examples suffice to demonstrate that in principle model theoretic definitions can be given for the various entailment relations introduced. As these definitions are, however, not used in the sequel, we omit the definitions of the other ones.

4.9 Fibred Semantics

As our labelled logics are a special case of LDS, one could argue that the general combination methods for labelled deduction systems (Gabbay, 1994*a*) can be applied. In this section we show that this is indeed the case.

Let us have a look at definition 4.8.1. If we consider the case that BL is also classical propositional logic, then we can combine the two (disjoint!) interpretations for labels and formulae into one single interpretation, taking the ":" operator as material implication, so that $\alpha{:}F$ becomes $\alpha \to F$. This does not change anything.

Following Gabbay one could do the same if BL were a logic different from classical propositional logic. The method proposed there, called *fibering*, says that in principle all one has to do is to define a language that contains the union of the connectives of the original languages and when interpreting a concrete formula by decomposing its structure to always use an interpretation of *that* logic that is responsible for the respective top level constructor.

For details we refer to Gabbay (1994*a*), but we want to give some additional notes: First, our case can be handled relatively easy, because our formulae have a particular structure. The components of the two logics are not interleaved arbitrarily, but there is only one single "$\to$", which is always the top level connective, connecting the otherwise separate parts of the formula. So we allow only a subset of the formulae we would get by fibering the two logics. This will be relaxed a little in section 8, when we allow for variables common to labels and formulae.

Second, we do not even *want* the general case, as the separation between the two parts of the combined formulae reflects the different components of a combined system. When we for instance talk about completeness, we are not interested in really considering all LL-entailed formulae, but only how a given formula part can be derived. Depending on the application the label part will not even be visible to the user (cf. chapter 7.5).

Third, we shall meet the same phenomenon, when we examine Poole's approach for default reasoning in chapter 8. There the *names* of defaults play the same rôle as our labels. In his translation also names imply the defining formula.

Chapter 5

Modelling Systems

In this chapter we show how to represent some of the known Reason Maintenance systems within our formalism. We shall not do this exhaustively, but simply demonstrate how the different characteristics can be modelled.

More importantly, this framework facilitates devising new systems by putting together the respective properties, in addition to simply describing and comparing known systems.

5.1 The ATMS of de Kleer

Easiest to be modelled is de Kleer's ATMS (de Kleer, 1986*a*), as our approach is essentially assumption-based as well. Besides the ATMS has already been given a rather simple semantics by Fujiwara & Honiden (1990) (see section 6.1).

The language of ATMS is propositional. Justifications are in essence Horn formulae, $A_1 \wedge A_n \rightarrow B$ expressing that B depends on the validity of all the A_i. Of course there may be more than one justification for an atom. Some of the atoms are special. They are called *assumptions*, and they are considered the axiomatic bases of dependency chains.

The similarity between ATMS labels and ours is obvious. If we apply the characteristic function χ (cf. definition 4.1.9) to a label, we get a collection of sets of formulae. This is exactly the same as an ATMS label tells us.

We translate an ATMS into a semi-basic set of labelled formulae as follows:

- every ATMS node corresponds to a classical propositional logic atom.
- every assumption is translated into a labelled formula $\alpha{:}A$ where A is the respective atom and α a "new" atomic label.
- every justification is translated to a formula $A_1 \wedge \ldots \wedge A_n \rightarrow B$, where B is the consequent and the A_i are the atoms corresponding to the antecedent nodes.
- the contradiction node is mapped to the falsum ($\perp$).
- nogood declarations are treated like any other justification.

The formula part of the resulting set of labelled formulae consists of formulae that are either Horn or positive (nogood justifications).

More formally we get

Definition 5.1.1 (Labelled ATMS Representation)
Let DN = (N, J, A, $\perp$) be an ATMS. Then we define injective functions that map elements of N$\setminus\perp$ to propositional atoms from BL and the logic of the labels, respectively. For $N \in$ N we denote the image as $\widetilde{N}$ and α_N and call it the **corresponding atom** resp. the **corresponding label** to N.

Then the **labelled representation** of DN is a set $\mathrm{LL}_{\mathrm{ATMS}}(\mathrm{DN})$ of labelled formulae, where classical propositional logic is the basic logic:

$$\mathrm{LL}_{\mathrm{ATMS}}(\mathrm{DN}) = \bigcup_{N \in \mathrm{A}} \mathrm{LL}_{\mathrm{ATMS}}(N) \cup \bigcup_{j \in \mathrm{J}} \mathrm{LL}_{\mathrm{ATMS}}(j)$$

where

$$\forall N \in \mathrm{A} \quad \mathrm{LL}_{\mathrm{ATMS}}(N) = \alpha_N{:}\widetilde{N}$$

$$\forall j \in \mathrm{J} \text{ with } \mathsf{conseq}(j) \neq \perp \quad \mathrm{LL}_{\mathrm{ATMS}}(j) = \top : \bigwedge_{\substack{N \in \\ \mathsf{inset}(j)}} \widetilde{N} \rightarrow \widetilde{\mathsf{conseq}}(j)$$

$$\forall j \in \mathrm{J} \text{ with } \mathsf{conseq}(j) = \perp \quad \mathrm{LL}_{\mathrm{ATMS}}(j) = \top : \bigvee_{N \in \mathsf{inset}(j)} \neg\widetilde{N}.$$

We note that ATMS assumptions become exactly our assumptions in the resulting semi-basic set, whereas justifications (ordinary and nogood-justifications) become non-assumptions.

The fundamental theorem for our translation from ATMS to sets of labelled formulae is theorem 4.1.15, which states that the above outline of a translation is in fact correct.

Given the work already done in chapter 4 the fundamental result can be obtained rather trivially:

Theorem 5.1.2 (ATMS Representation Theorem)
Given an ATMS DN = (N, J, A, $\perp$) and its representation as a set $\Phi = LL_{\mathrm{ATMS}}(DN)$ of labelled formulae according to definition 5.1.1. Then for any node $N \in$ N holds: If l is the label computed for N by the ATMS procedure, then the characteristic formula of l is logically equivalent to the maximal label for which $\Phi \models_{\mathrm{consLL}} \widetilde{N}$ holds.

Proof:
We presuppose, that the label computed by the ATMS is consistent, sound, complete and minimal. According to de Kleer it always exists and is unique. If we can show that the characteristic function of the maximal label fulfills the requirements, we are done because of the uniqueness. Let α be the maximal label for which $\Phi \models_{\mathrm{consLL}} \widetilde{N}$ holds.

1. $\chi(\alpha)$ is sound: because of the maximality α must be positive and relevant. Therefore $\chi(\alpha)$ exists. With theorem 4.1.15 we obtain the result.

2. $\chi(\alpha)$ is consistent: from the definition of $\models_{\text{consLL}}$

3. $\chi(\alpha)$ is complete: from the maximality of α

4. $\chi(\alpha)$ is minimal: this follows directly from the DDNF form. □

5.2 Doyle's TMS

In the TMS there is no distinguished class of nodes serving as assumptions. However, there are some *implicit assumptions* present, namely *every node can simply be assumed* `out` if there is no justification for the contrary. The introduction of these as explicit assumptions gives us the possibility to treat the TMS similarly to the ATMS.

The general idea is to model justifications by implications, as it was done in the ATMS case. A justification with consequent C, IN-SET $\{A\}$ and OUT-SET $\{B\}$ would then simply become $\top{:}A \wedge \neg B \rightarrow C$. Formulae of type $\alpha{:}F$ with atomic F are not introduced this time, as there are no distinguished items that are explicitly marked as assumptions. Instead we add $\alpha{:}\neg F$ (with fresh α) for *all* nodes (atomic formulas) introduced, which represents the implicit assumptions.

We have to be careful in modelling the justifications, though, because in the TMS IN-SET and OUT-SET are not handled symmetrically. The crucial point to be represented is the notion of *well-foundedness*. This has to do with the directionality of justifications. It should not be possible to deduce backward from an `out` consequence, that an antecedent in the OUT-SET is `in`, for this violates well-foundedness. This can be blocked by replacing the justification $\top{:}A \wedge \neg B \rightarrow C$ from above by $\beta{:}A \wedge \neg B \rightarrow C$, where $\beta{:}\neg B$ was the assumption that B is `out`. More generally: we take as label for the justification formulae the conjunction of the labels of its out-antecedents.

Besides, the TMS does not yield enumerations of environments (labels) for every node considered in isolation, but computes one (of possibly many, if one exists at all) *labelling*, representing a global state. We capture this by the input completeness demand.

Definition 5.2.1 (Labelled TMS Representation)
Let DN $= (N, J, E)$ be a TMS. Now we define two injective functions, that map every node $A \in N$ to a BL-atom $\widetilde{A}$, the so called **corresponding atom**, and the PL-atom α_A, the **corresponding label**, respectively. The (partial) inverse functions yield A as the **corresponding node**.

Then the **labelled representation** of DN is a set $\mathrm{LL_{TMS}(DN)}$ of labelled formulae over classical propositional logic as basic logic, obtained as

$$\mathrm{LL_{TMS}(DN)} = \bigcup_{N \in \mathrm{N}} \mathrm{LL_{TMS}}(N) \cup \bigcup_{j \in \mathrm{J}} \mathrm{LL_{TMS}}(j)$$

where

$$\begin{array}{lll} \forall N \in \mathrm{N} & \mathrm{LL_{TMS}}(N) & = \alpha_N{:}\neg\widetilde{N} \\ \forall j \in \mathrm{J} & \mathrm{LL_{TMS}}(j) & = \prod_{\substack{A\in\\ \mathsf{outset}(j)}} \alpha_A \;:\; \big(\bigwedge_{\substack{A\in\\ \mathsf{inset}(j)}} \widetilde{A} \;\wedge\; \bigwedge_{\substack{A\in\\ \mathsf{outset}(j)}} \neg\widetilde{A}\big) \rightarrow \widetilde{\mathsf{conseq}}(j). \end{array}$$

This may seem a little odd on first view. What happens is the following: contrapositives are not suppressed, but using the implication representing the justification in the contrapositive direction also adds the respective label to the respective context. This, however, results in a nogood, as the out-antecedent concerned is justified as being `out` with the same label.

In our definition of the labelled representation contradiction nodes are not handled specially, but mapped to ordinary BL atoms. If we want to treat dependency directed backtracking or talk about correct nogood strategies in the Elkan sense, we need a different type of transformation that takes care of this.

Definition 5.2.2 (TMS Representation with Contradiction Nodes) Let $\mathrm{DN} = (N, J, E)$ be a TMS. The labelled representation of DN, **accounting for contradiction nodes**, is a set $\mathrm{LL_{TMS'}}(\mathrm{DN})$ of labelled formulae over classical propositional logic as basic logic, obtained as

$$\mathrm{LL_{TMS'}}(\mathrm{DN}) = \bigcup_{N\in\mathrm{N}\setminus\{\perp\}} \mathrm{LL_{TMS'}}(N) \cup \bigcup_{j\in\mathrm{J}} \mathrm{LL_{TMS'}}(j)$$

where

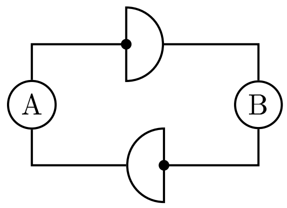
$$\begin{array}{llllll} \forall N \in \mathrm{N} & \text{with} & N \neq \perp & \mathrm{LL_{TMS'}}(N) & = & \mathrm{LL_{TMS}}(N) \\ \forall j \in \mathrm{J} & \text{with} & \mathrm{CONS}(j) \neq \perp & \mathrm{LL_{TMS'}}(j) & = & \mathrm{LL_{TMS}}(j) \\ \forall j \in \mathrm{J} & \text{with} & \mathrm{CONS}(j) = \perp & \mathrm{LL_{TMS'}}(j) & = & \top : \bigvee_{\substack{A\in\\ \mathsf{inset}(j)}} \neg\widetilde{A}. \end{array}$$

We can now try to find labels for the atomic formulae, that are *consistent* and *input complete*, i.e. this time we are interested in consequences with respect to $\models_{\max(\mathrm{cons},\mathrm{inputLL})}$. This will in fact give us a representation of all possible labellings. Of course the nondeterminism of TMS (which concrete labelling to choose) can not be resolved by our methods.

The case is more complicated than ATMS, because this time we have to start with sets of formulae that are not even semi-basic. However, they are simple.

Let us start with some examples. The notation we use for these is the following: first we give the set of labelled formulae the dependency net is mapped to. Then for every atom and its negation from the unlabelled set, as well as for $\perp$, we give the maximal label according to simple LL-entailment. Because of the "computation rules" presented in the preceding chapter we can immediately read off the maximal consistent and input complete labels from this representation.

Example 5.2.3 (Even Loop)



This will be modelled as

$$\alpha : \neg A \quad \beta : \neg B$$

$$\alpha + \beta : A \vee B$$

The second formula is simply a contraction of the two formulae $\alpha{:}A \vee B$ and $\beta{:}A \vee B$ which model the two nonmonotonic justifications.

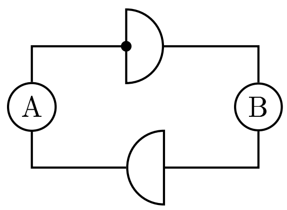
The resulting LL-consequences with maximal labels are

$$\begin{array}{ccc} \beta{:}A & & \alpha{:}\neg A \\ \alpha{:}B & & \beta{:}\neg B \\ & \alpha\beta{:}\bot & \end{array}$$

We are now looking for the maximal consistent label which is input complete, i.e. its prime implicants make either A or $\neg A$ true for every atom A corresponding to a node. Here this label is $\alpha + \beta$. To see this we can look at the table of maximal labels for the respective nodes and immediately see that either α or β is needed to guarantee input completeness. Since $\alpha + \beta$ contains no nogood (no prime implicant implies the maximal label for $\bot$), we are done.

What is the intuitive meaning of this? Remember labels stand for sets of sets of assumptions. In TMS the assumptions are the implicit assumptions of nodes being `out`. Here are two environments, characterized by α and β. This means there are two environments which give us a complete well-founded labelling, namely assuming A to be `out` or assuming B to be `out`, which certainly meets our intuition. This is very much alike extensions as defined in default logics (cf. chapter 8).

Example 5.2.4 (Odd Loop)



This is modelled as

$$\alpha : \neg A \qquad \beta : \neg B$$

$$\alpha : A \vee B$$

$$\top : A \vee \neg B$$

The last two formulae can be replaced by $\top{:}A \vee \neg B$, as this subsumes the other one.

This time the consequences with maximal labels are

$$\begin{array}{lcr} \alpha{:}A & & \alpha{:}\neg A \\ \alpha{:}B & & \alpha + \beta{:}\neg B \\ & \alpha{:}\bot & \end{array}$$

The shaded parts of the formulae are nogoods. Since α is a nogood, there is no consistent input complete label, because it is the maximal label for A as well as for $\neg A$. This corresponds to the nonexistence of a well-founded labelling.

Before we can proceed to the main theorem of this section, we have to introduce some terminology.

Definition 5.2.5 (Stable Labels)
A consistent, input complete label β which is a conjunction of atoms, containing all the atoms α_A for which $\Phi \models_{\mathrm{LL}} \beta{:}\neg\widetilde{A}$ holds, is called **stable**.

Definition 5.2.6 (Canonical Label Transformation)
Let DN be a TMS and $\mathrm{LL}_{\mathrm{TMS}}(\mathrm{DN})$ its labelled representation. Then any labelling l of DN can be mapped to the **canonical label** $\lambda(l)$, which is defined as

$$\lambda(l) = \prod_{l(N)=\mathtt{out}} \alpha_N$$

Since λ is injective (up to ordering), there is a (partial) inverse function λ^{-1} that maps labels consisting of conjunctions of atomic labels that are corresponding to nodes of DN to labellings of DN.

Lemma 5.2.7
λ^{-1} is always defined on stable labels.

Proof:
A stable label α is by definition input complete and consistent. So for every node A we have either $\Phi \models_{\mathrm{LL}} \alpha{:}\widetilde{A}$ or $\Phi \models_{\mathrm{LL}} \alpha{:}\neg\widetilde{A}$ (exclusive). □

We can now approach the fundamental theorem of JTMS representation: it shows that there is a one to one correspondence between the sound and well-founded labellings of a TMS and the stable labels of its labelled representation. We prove this in two steps.

Proposition 5.2.8
Let DN = (N, J, E) be a TMS, and $\Phi = LL_{\mathrm{TMS}}(DN)$ its labelled representation according to definition 5.2.1. Let further l be a sound and well-founded labelling of DN. Then the canonical label $\lambda(l)$ is stable.

Proof:
α trivially fulfills the first criterion for being stable, namely containing exactly the labels corresponding to nodes labelled `out`, by definition of the canonical label. We have yet to prove that it is input complete and consistent.

First we show that the label is input complete. We consider the nodes labelled `out` and the ones labelled `in` by l separately.

1. For all nodes A labelled `out` of course $\Phi \models_{\text{LL}} \alpha{:}\neg\widetilde{A}$ holds because we have $\text{LL}_{\text{TMS}}(\text{A}) = \alpha_A{:}\neg\widetilde{A} \in \Phi$ and α_A is a literal in α.

2. There exists a function rank, such that, if a node A is labelled `in`, then there exists a valid justification for A with the rank of all nodes in its IN-SET smaller than the rank of A. We use this function for an induction to prove $\Phi \models_{\text{LL}} \alpha{:}\widetilde{A}$ for all the nodes A labelled `in`:

   The statement holds for all A with the lowest occurring rank, for the only possibility for them are justifications with empty IN-SET. The way justifications are translated we have a formula

   $$\beta{:}\widetilde{A}_1 \vee \ldots \vee \widetilde{A}_n \vee \widetilde{A}$$

   where β is a conjunction of atomic labels. Since the justification is valid, all the A_i are `out` and therefore their corresponding labels appear in α. We also have $\Phi \models_{\text{LL}} \alpha{:}\neg\widetilde{A}_i$, so we can conclude $\Phi \models_{\text{LL}} \alpha{:}\widetilde{A}$. Now we assume this holds for all A with a rank smaller than some given value. If we consider a node labelled `in` with the next rank occurring we know that there must be a valid justification for this node with all the nodes in the IN-SET having a smaller rank. Therefore we can assume they are all α-implied. With this we can perform the induction step, concluding that the statement indeed holds for all nodes labelled `in`.

Now we show that such a label is consistent. Suppose there is a BL-formula F with $\Phi \models_{\text{LL}} \alpha{:}F$ and $\Phi \models_{\text{LL}} \alpha{:}\neg F$. Then there must be a BL-inconsistent subset of formula(Φ) with a label that PL-implies α. We show that the set of all formulae in Φ with a label implying α is BL-consistent. So must every subset be, and we get a contradiction.

We construct this set Ψ by including all the formulae from Φ labelled $\top$, plus those whose labels contain only literals occurring in α. This is the biggest subset of Φ with a label implying α. We now construct a BL-interpretation $\Im$ by defining

$$\forall A \in N \quad \Im \models_{\text{BL}} \widetilde{A} \quad \text{iff} \quad l(A) = \texttt{out}.$$

We claim that $\Im \models_{\text{BL}}$ formula(φ) for all formulae φ in Ψ. This is easy to prove: If $\varphi = \text{LL}_{\text{TMS}}(j)$ for some justification j, then j is valid if one of the IN-antecedents is `out`, one of the OUT-antecedents is `in` or the consequent is `in`. The OUT-antecedents are all `out` in our construction. But since the labelling is sound, the consequent must be `in` if none of the IN-antecedents is `out`. So our interpretation $\Im$ must satisfy the corresponding formulae also. If $\varphi = \text{LL}_{\text{TMS}}(N)$ for some node N, it is trivially satisfied because of the definition of $\Im$. So there exists a BL-model $\Im$ of $\mathsf{formula}(\Psi)$. This means this set and all its subsets are BL-consistent. □

We now come to the converse direction.

Proposition 5.2.9
Let DN = (N, J, E) be a TMS, and $\Phi = LL_{\mathrm{TMS}}(DN)$ its labelled representation, and let α be a stable label of Φ. Then $\lambda^{-1}(\alpha)$ is sound and well-founded.

Proof:
Because of lemma 5.2.7 we know that $\lambda^{-1}(\alpha)$ is defined. Now consider the labelling $\lambda^{-1}(\alpha)$. We have to show that this labelling is indeed sound and well-founded.

We start with soundness. A justification with IN-antecedents $A_1, \ldots, A_n$, OUT-antecedents $B_1, \ldots, B_m$ and consequent C is translated to

$$\beta_{B_1} \ldots \beta_{B_m}{:}\neg\widetilde{A}_1 \vee \ldots \vee \neg\widetilde{A}_n \vee \widetilde{B}_1 \vee \ldots \vee \widetilde{B}_m \vee \widetilde{C}.$$

If all the IN-antecedents are `in` and all the OUT-antecedents `out`, this means $\Phi \models_{\mathrm{LL}} \alpha{:}\widetilde{A}_i$ and $\Phi \models_{\mathrm{LL}} \alpha{:}\neg\widetilde{B}_i$, we also have $\Phi \models_{\mathrm{LL}} \alpha{:}\widetilde{C}$, because all the β_{B_i} are contained in α.

We now show how to construct the rank function for the proof of well-foundedness.

For this purpose we take a look at the atoms A for which $\Phi \models_{\mathrm{LL}} \alpha{:}\widetilde{A}$ holds. From the definition of $\models_{\mathrm{LL}}$ we know that there must exist collections of subsets of Φ whose formula parts BL-entail $\widetilde{A}$ and the disjunction of whose labels is PL-implied by α. Because of the construction of α as conjunction of atoms and the fact that Φ is simple this can be strengthened: there must exist a subset Ψ of Φ with $\mathsf{formula}(\Psi) \models_{\mathrm{BL}} \widetilde{A}$ and $\models_{\mathrm{PL}} \alpha \rightarrow \mathsf{label}(\Psi)$. Now consider $\Psi' = \chi(\mathsf{label}(\Psi))$ instead. Since it is clearly the case that $\Psi' \supseteq \Psi$, we get $\mathsf{formula}(\Psi') \models_{\mathrm{BL}} \widetilde{A}$ as well. Furthermore Ψ' can not be BL-inconsistent, because this would contradict the consistency of α, since $\mathsf{label}(\Psi') = \mathsf{label}(\Psi)$. So $\mathsf{formula}(\Psi')$ has BL-models.

How do these BL-models look? In Ψ' there are three types of formulae:

- $\alpha_{A_i}{:}\neg\widetilde{A}_i$ with atomic label α_{A_i} occurring in α
- possibly justifications of contradiction nodes of the form

$$\top{:}\neg\widetilde{A}_1 \vee \ldots \vee \neg\widetilde{A}_n$$

- translations of ordinary justifications of the form

$$\beta_{B_1} \ldots \beta_{B_n}{:}\neg\widetilde{A}_1 \vee \ldots \vee \neg\widetilde{A}_n \vee \widetilde{B}_1 \ldots \vee \widetilde{B}_m \vee \widetilde{C}.$$

If there are no formulae of the third kind, then $\mathsf{formula}(\Psi')$ does not BL-entail any positive atom, for the interpretation mapping all atoms to `false` is clearly a BL-model of $\mathsf{formula}(\Psi')$. Because of the formulae of the first type every model of $\mathsf{formula}(\Psi')$ has to map the respective atoms to `false`. These appear again as the B_i in the third kind of formulae. Since all the β_i appear in α, all the B_i are mapped to `false` in a model of $\mathsf{formula}(\Psi')$ and thus the respective atoms do not contribute to satisfying the respective formula.

The only possibility to force an atom to be present (positively) in a model is the existence of a formula of the third kind with no negative literals. This shows that the consequent $\widetilde{C}$ is BL-entailed. This corresponds exactly to a valid justification with no IN-antecedents. We can start the definition of rank by giving rank 0 to all the nodes C for which this is the case.

The fact that these atoms are present in all models of $\mathsf{formula}(\Psi')$ forces some others to be present also, iff there exist formulae of the third kind which contain as negative literals only literals of rank 0. There also the consequents have to be in the model. We rank them with 1.

We can proceed this way until there is no formula left which contains as negative literals only literals of preceding ranks. All remaining nodes can be assumed to be falsified by an interpretation without any problem and thus are not BL-entailed.

Of course Ψ' may have fewer models than Ψ, so Ψ' could BL-entail more formulae. But this is not critical, since the pure *existence* of a set like Ψ also guarantees the existence of Ψ'. □

The main result of this section is now a corollary of the two preceding propositions:

Corollary 5.2.10 (TMS Representation Theorem 1)
Let DN = (N, J, E) be a TMS, and $\Phi = LL_{\mathrm{TMS}}(DN)$ its labelled representation. Then there is a bijection between the stable labels of Φ and the canonical labels of sound and well-founded labellings of DN.

Corollary 5.2.11
Let Φ be the labelled representation of the dependency net DN. Then there exists a sound, well-founded labelling for DN, iff Φ possesses a stable label.

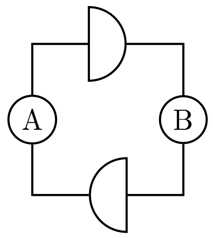
An interesting question is how to compute the stable labels and thus the labellings of a dependency net. The general procedure is first to compute the maximal consistent, input satisfying label. This can be done in steps, first computing the maximal (LL) labels for every atom and its negation, as well as for the falsum, combining these in the appropriate way.

Then one looks at the prime implicants of this label. Since the maximal label is positive, so are the prime implicants. Furthermore every prime implicant is input complete. Since every input complete label is also input satisfying, any input complete label implies the maximal label and in particular any input complete conjunction of literals implies one of the prime implicants, i.e. it has at least the same atoms.

Now one has to check stability. For this we add to every prime implicant all the atomic labels corresponding to nodes whose negation is LL-entailed with this label. Some of the prime implicants will then yield labels which are not consistent. Also several of the prime implicants may lead to the same label.

This is illustrated by two examples:

Example 5.2.12
The net



is translated to

$$\alpha : \neg A \qquad \beta : \neg B$$

$$\top : \neg A \vee B$$

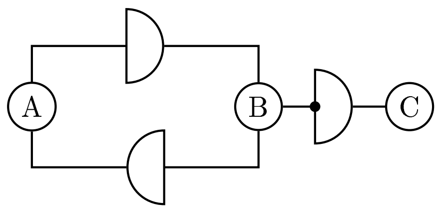
$$\top : A \vee \neg B$$

resulting in the following maximal labels:

$$\begin{array}{lcl} \bot{:}A & & \alpha+\beta{:}\neg A \\ \bot{:}B & & \alpha+\beta{:}\neg B \\ & \bot{:}\bot & \end{array}$$

The candidate labels are α and β, but they both are mapped to the same stable label, $\alpha\beta$, which corresponds to the only sound and well-founded labelling for this dependency net, namely both nodes labelled `out`.

Example 5.2.13
The net



translates to

$$\alpha : \neg A \qquad \beta : \neg B \qquad \gamma : \neg C$$

$$\top : \neg A \vee B$$

$$\top : A \vee \neg B$$

$$\beta : B \vee C$$

This time we give the maximal labels with nogoods already eliminated:

$$\begin{array}{ccc} \bot{:}A & & \alpha+\beta{:}\neg A \\ \bot{:}B & & \alpha+\beta{:}\neg B \\ \beta{:}C & & \gamma{:}\neg B \\ & \beta\gamma{:}\bot & \end{array}$$

So the maximal consistent, input complete label is $\alpha\gamma + \beta$. Whereas β can be expanded to the stable $\alpha\beta$, the label $\alpha\gamma$ (which makes all nodes `out`, which is not a sound labelling) can not be enlarged by adding β without losing consistency, though $\Phi \models_{\mathrm{LL}} \alpha\gamma{:}\neg B$.

In our representation theorem we have ignored the particular semantics of contradiction nodes, since the translation according to definition 5.2.1 treats them as ordinary nodes and hence does not force them to be labelled `out`. Using definition 5.2.2 instead, we can overcome this flaw and get a stronger version of the TMS representation theorem:

Theorem 5.2.14 (TMS Representation Theorem 2)
Let $DN = (N, J, E)$ be a TMS, and $\Phi = LL_{\mathrm{TMS}'}(DN)$ its labelled representation (accounting for contradiction nodes). Then there is a bijection between the stable labels of Φ and the canonical labels of sound and well-founded labellings of DN that label contradiction nodes `out`.

Proof:
The proof follows in essence the proofs for propositions 5.2.8 and 5.2.9. The difference is, that this time contradiction nodes have no direct atomic counterpart. $\mathrm{LL}_{\mathrm{TMS}'}$ differs from $\mathrm{LL}_{\mathrm{TMS}}$ only in this respect. Let $\mathrm{LL}_{\mathrm{TMS}}$ map a contradiction node to an atom C. Then a labelling assigns the label `in` to this contradiction node, if the corresponding label α is a label for C, i.e. $\alpha{:}C$ holds. Looking at the $\mathrm{LL}_{\mathrm{TMS}'}$ translation this means $\alpha{:}\bot$ holds, which simply means α is inconsistent. So labellings that map a contradiction node to `in` are excluded by $\mathrm{LL}_{\mathrm{TMS}'}$ because of the consistency criterion. □

It is important to emphasize that our modelling of the TMS has a one to one correspondence between the TMS's justifications and labelled formulae. Therefore the translation is purely local, and it is possible to extend such a model by further formulae if justifications are added to the TMS. This is in contrast to characterizations like that of Junker & Konolige (1990*a*), which maps labellings (taking the whole dependency net into consideration for the translation of every single justification) and therefore cannot be extended that simply. They have to completely restart the whole procedure for the addition of a justification. In our case, however, the logical representation can be incrementally updated very much the same way as the original net.

5.2.1 Dependency directed Backtracking

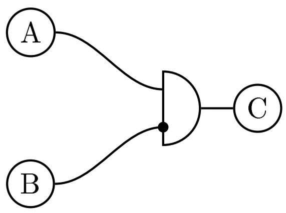
Though we have accounted for contradiction nodes in the preceding section, we have not really modelled the dependency directed backtracking procedure. Our translation of contradiction nodes guarantees that they are never labelled `in`. The reason for this is that contradiction nodes do not show up as separate atoms, like all the other nodes, that can be freely interpreted, but are directly mapped to the falsum. So we have not treated dependency directed backtracking in the way the TMS treats the matter. Although we believe our approach is an improvement, because it pursues a correct nogood strategy, one may want to exactly imitate what the TMS does. There are two possibilities for this.

1. The first is to use the translation from definition 5.2.1 instead of the one in definition 5.2.2. Then one only has theorem 5.2.10 instead of theorem 5.2.14 with the effect that contradiction nodes may be labelled `in`. Now one could take an operational view and just enlarge the set of formulae by one that represents the new justification introduced by the TMS. This is certainly correct, but not very satisfactory, since that does not supply any semantics.

2. The second possibility is motivated by what DDB was originally intended for. It introduces new justification possibilities by allowing to infer backward from contradictions in certain cases. This can be done by changing the logic of the labels in such a way that justifications are represented by an appropriate kind of implication. This is certainly the most interesting variant, but it is beyond the scope of this thesis.

3. The approach we shall pursue within this section is to allow restricted contrapositive use of justifications by adding extra labels to them. We give some examples of how this works, but unfortunately we are not able to give characterizing theorems, as the approach does more than the original DDB. Therefore this section stays rather informal.

Have a look at the example of a single justification



We used to model this with the formulae

$$\alpha : \neg A \qquad \beta : \neg B \qquad \gamma : \neg C$$

$$\beta : A \wedge \neg B \to C$$

The PL-implication can be used in three directions, only one of which is usually wanted. Let us check what happens:

1. If we have a reason δ for A being `in`, i.e. $\delta{:}A$ holds, then we can then use the formula describing the justification above to derive $\beta\delta{:}C$. This is the direction we normally want to use.

2. If we instead have reasons for C being `out` ($\delta{:}\neg C$) and A being `in` ($\epsilon{:}A$), we can work in the backwards direction to conclude $\beta\delta\epsilon{:}B$. But we also have $\beta{:}\neg B$. Therefore $\beta\delta\epsilon$ is a nogood and this inference is blocked.

3. The last direction thinkable is deriving $\beta\delta{:}\neg A$ from $\delta{:}\neg C$ (and $\beta{:}\neg B$). But this is not really new, for we can deduce $\alpha{:}\neg A$ anyway. The two labels do not imply one another, but if α turns out to be a nogood, then of course $\beta\delta$ is a nogood, too, for the same derivation for $\perp$ could be used simply starting with $\beta\delta{:}\neg A$ instead of $\alpha{:}\neg A$.

So the ordinary mechanism (without DDB) works as expected. The "wrong" directions are excluded by labelling the justification not with $\top$ but with the conjunction of all antecedents in the OUT-SET.

For modelling DDB we *want* to be able to use justifications backward. If in the example C constitutes a nogood, we want to not only force C to be `out`, but use this as a reason for either A being `out` or B being `in`. Already here it becomes apparent that we do not want to copy the faults accompanying DDB: The normal DDB procedures will try to make B `in`, neglecting the possibility of alternative labellings with A labelled `out`. As already noted, there are possibilities to model the multidirectionality by alternative definitions of the implication and therefore digression from PL as logic for the labels. One could for example use three-valued logics to be able to discriminate between "`out` because of lack of justifications for being `in`" and "*must necessarily* be `out`". Here we go a simpler way by allowing for certain backward directions by changing the label for the justification from β to $\beta + \overline{\gamma}$, i.e. permitting alternative derivations that do not include the β. The new justification

$$\beta + \overline{\gamma} : A \wedge \neg B \rightarrow C$$

should be understood as saying "in order to use me, you either have to assume that B is `out`, or you may *not* use the assumption that C is `out`". Why this? We want to be able to deduce backwards from C's state of being *necessarily* `out` (because of nogood justifications), but this should not be possible simply from the *assumption* that C is `out`, which is introduced by $\gamma{:}\neg C$.

Let us revisit our example with the new translation:

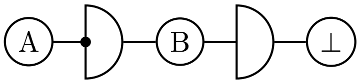
1. Given $\delta{:}A$ and $\epsilon{:}\neg B$ we now can conclude $\beta\delta\epsilon + \overline{\gamma}\delta\epsilon{:}C$.

2. If we instead have $\delta{:}\neg C$ and $\epsilon{:}A$, we get $\beta\delta\epsilon + \overline{\gamma}\delta\epsilon{:}B$. The first part $\beta\delta\epsilon$ is again a nogood, but $\overline{\gamma}\delta\epsilon{:}B$ still remains. Note that any environment containing this cannot also contain the *assumption* that C is `out`, because $\overline{\gamma}\delta\epsilon$ and γ are contradictory.

3. From $\delta{:}\neg C$ and $\epsilon{:}\neg B$ we get $\beta\delta\epsilon + \overline{\gamma}\delta\epsilon{:}\neg A$.

In order to demonstrate that this translation works as expected, we give some examples.

Example 5.2.15



Our first translation (without DDB-correction)

$$\alpha : \neg A \qquad \beta : \neg B$$

$$\alpha : A \vee B$$

$$\top : \neg B$$

yields

$$\begin{array}{ccc} \alpha{:}A & & \alpha{:}\neg A \\ \alpha{:}B & & \top{:}\neg B \\ & \alpha{:}\bot & \end{array}$$

This means there is no consistent input complete label and therefore no complete and well-founded labelling that assigns `out` to the contradiction node. The task of DDB is to find a culprit for the inconsistency. Here B is the only candidate. In order to defeat it, there is again only one choice: A must be supplied with a valid justification. This is exactly what the TMS would do by adding a justification for A with B as only (IN-)antecedent.

Instead of adding a justification, we allow using the justifications backwards. The fact that $\bot$ *must* be `out` justifies B being `out` and this in turn gives a justification for A being `in`.

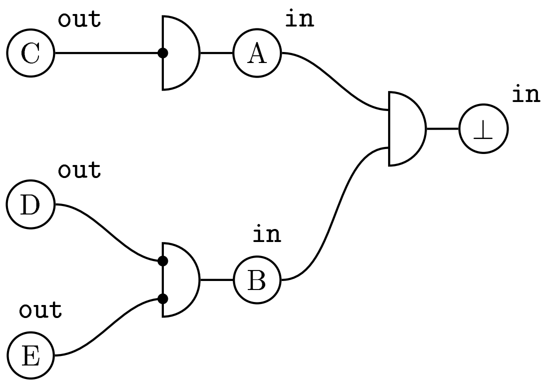
Therefore we replace $\alpha : A \vee B$ by $\alpha + \overline{\beta} : A \vee B$, thus obtaining as maximal labels

$$\begin{array}{ccc} \alpha + \overline{\beta}{:}A & & \alpha{:}\neg A \\ \alpha{:}B & & \top{:}\neg B \\ & \alpha{:}\bot & \end{array}$$

Now there is a consistent input complete label, namely $\overline{\beta}$. In TMS terms that means that the fact that it is forbidden to assume B `out` serves as a justification for A being `in`.

Our procedure is best suited to model a cautious type of DDB that does not label nodes `in` unnecessarily. The next example is already known from chapter 2 (example 2.2.5).

Example 5.2.16



Without the DDB correction this dependency net is modelled as

$$\alpha : \neg A \qquad \beta : \neg B \qquad \gamma : \neg C$$

$$\delta : \neg D \qquad \epsilon : \neg E$$

$$\gamma : A \vee C$$

$$\top : \neg A \vee \neg B$$

$$\delta\epsilon : B \vee D \vee E.$$

As a result the following maximal consistent labels are obtained:

$$\begin{array}{rr}
\alpha + \delta\epsilon{:}\neg A & \gamma{:}A \\
\beta + \gamma{:}\neg B & \delta\epsilon{:}B \\
\gamma{:}\neg C & \text{(none)}{:}C \\
\delta{:}\neg D & \text{(none)}{:}D \\
\epsilon{:}\neg E & \text{(none)}{:}E \\
\multicolumn{2}{c}{\gamma\delta\epsilon + \beta\delta\epsilon + \alpha\gamma{:}\perp}
\end{array}$$

There is no input complete label, and thus no extension.

With DDB modelling via reverse justifications, the formulae change to

$$\alpha : \neg A \qquad \beta : \neg B \qquad \gamma : \neg C$$

$$\delta : \neg D \qquad \epsilon : \neg E$$

$$\gamma + \overline{\alpha} : A \vee C$$

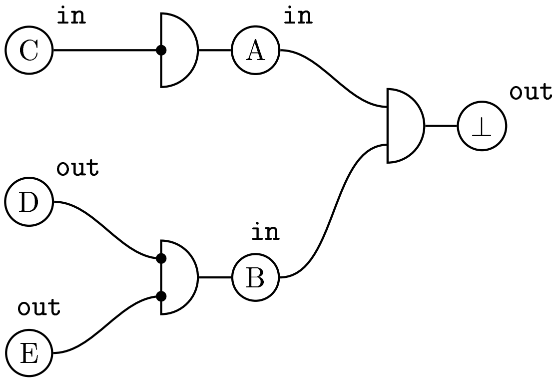
$$\top : \neg A \vee \neg B$$

$$\delta + \overline{\beta}\epsilon : B \vee D \vee E,$$

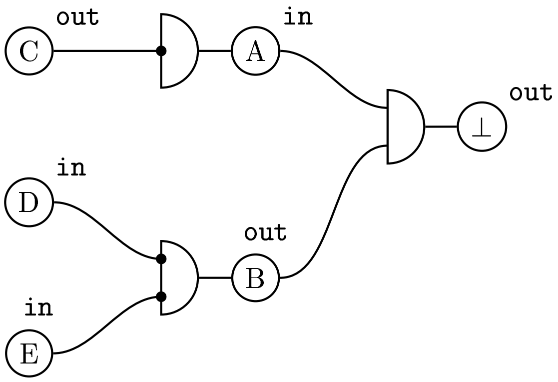
yielding

$$\begin{array}{rr} \alpha + \delta\epsilon{:}\neg A & \gamma{:}A \\ \beta + \gamma{:}\neg B & \delta\epsilon{:}B \\ \gamma{:}\neg C & \overline{\alpha}\delta\epsilon{:}C \\ \delta{:}\neg D & \overline{\beta}\gamma\epsilon{:}D \\ \epsilon{:}\neg E & \overline{\beta}\gamma\delta{:}E \end{array}$$
$$\gamma\delta\epsilon + \beta\delta\epsilon + \alpha\gamma{:}\perp$$

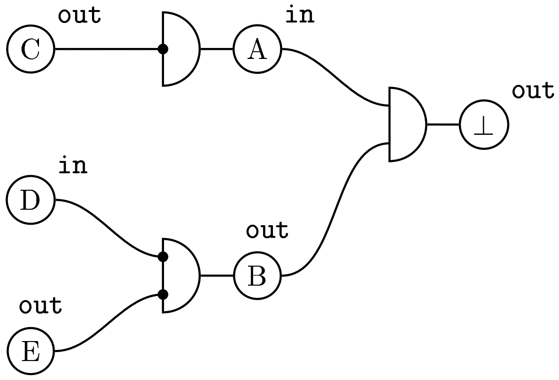
This time the maximal input complete consistent label is $\overline{\alpha}\delta\epsilon + \overline{\beta}\gamma\delta + \overline{\beta}\gamma\epsilon$, leading to three possible extensions, namely the one characterized by assuming D and E `out` ($\delta\epsilon$):



the one obtained by assuming B and C `out` ($\beta\gamma$):



and the one given by assuming C and E `out` ($\gamma\epsilon$):



Note that the label $\gamma\epsilon$ is also a label for $\neg B$, so in order to check for stability, we must test whether $\beta\gamma\epsilon$ is a nogood, which is not the case. Note that the negative atoms occurring in the labels are only needed to exclude some unwanted labels. They have to be ignored in the stability check.

Adding $\top{:}E$ in this situation yields

$$\begin{array}{rr} \alpha{:}\neg A & \gamma{:}A \\ \beta+\gamma{:}\neg B & \text{(none)}{:}B \\ \gamma{:}\neg C & \text{(none)}{:}C \\ \delta{:}\neg D & \text{(none)}{:}D \\ \epsilon{:}\neg E & \top{:}E \end{array}$$
$$\epsilon + \alpha\gamma{:}\bot$$

and thus simply chooses the third labelling as the only remaining option. Note that D is `out`.

As a further example let us take the Nixon Diamond example (Poole, 1985). This shows how careful our description of DDB proceeds. This is taken from a default logic formulation. Q, R, D, H, AQ, AR stand for the predicates QUAKER, REPUBLICAN, DOVE, HAWK, ABNORMAL-QUAKER, respective ABNORMAL-REPUBLICAN.

Example 5.2.17 (Nixon Diamond)
Directly translating with the DDB-correction we get

$$\alpha+\overline{\epsilon} : \neg Q \vee AQ \vee D \quad \beta+\overline{\eta} : \neg R \vee AR \vee H$$

$$\begin{array}{lll} \top : \neg D \vee \neg H & \top : Q & \top : R \\ \alpha : \neg AQ & \beta : \neg AR & \gamma : \neg Q \\ \delta : \neg R & \epsilon : \neg D & \eta : \neg H \end{array}$$

with

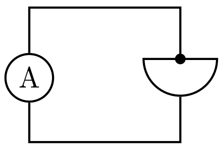
$$\begin{array}{rr}
\alpha\beta + \beta\overline{\epsilon} + \alpha\epsilon : AQ & \alpha : \neg AQ \\
\alpha\beta + \alpha\overline{\eta} + \beta\eta : AR & \beta : \neg AR \\
\top : Q & \gamma : \neg Q \\
\top : R & \delta : \neg R \\
\alpha : D & \beta + \epsilon : \neg D \\
\beta : H & \alpha + \eta : \neg H \\
\multicolumn{2}{c}{\alpha\epsilon + \beta\eta + \alpha\beta + \gamma + \delta : \bot}
\end{array}$$

as resulting maximal labels.

The maximal input consistent label is $\alpha\overline{\eta} + \beta\overline{\epsilon}$. This means there are two possible labellings which could result from DDB, both making Q and R `in`, but one assuming ABNORMAL-QUAKER `out` and then being forced not two assume HAWK `out`, thus making ABNORMAL-REPUBLICAN `out` and DOVE `in`, the other just the other way round. Here it is *not* decided which alternative is chosen by a nondeterministic implementation of DDB, which seems quite reasonable.

The case that is handled differently than by Doyle's DDB is the direct odd loop:

Example 5.2.18



Here the translation is

$$\alpha : \neg A$$

$\top$[1]:A

which results in

$$\alpha : \neg A \qquad\qquad \top : A$$
$$\alpha : \bot$$

which yields $\overline{\alpha}$ as maximal input consistent label. So we have a labelling that makes A `in`. On first view this is rather suspicious, since Doyle's TMS gives no labelling, and indeed ours is not well-founded. Furthermore there is no contradiction node present, so that there is no need for DDB. However, as shown by Elkan (1990), this odd loop can be viewed as equivalent to a contradiction node. Therefore we are quite content with this result.

Concluding, the author has to say that he is not really satisfied with the solution presented in this section. It should be much more interesting to examine altered label logics, as already hinted at.

[1]From $\alpha + \overline{\alpha}$

5.3 Generalized Sets and the System of Martins and Shapiro

We want to give an intuitive way of how to interpret labels containing negative literals. Our view of the "collection of sets" can indeed be generalized to contain this as well.

Let us remember the case of ATMS. We saw that maximal labels in an ATMS stand for sets of sets which entail a given proposition. As the ATMS is monotonic, every superset of a context also entails that proposition. So there is nothing wrong with assuming that labels represent the respective sets *and all their supersets.*

When we consider nogoods and therefore consistent LL-entailment the situation is different. The maximal consistent label again gives us environments that allow the deduction of the respective node. In addition these environments are guaranteed not to contain nogoods. But it is not the case that supersets of these environments are also free of nogoods. If we want to know of *all* environments that entail the node in question and besides are consistent, we cannot directly get them without considering the maximal label for $\perp$.

There is, however, an alternative way to represent nogoods, that can yield the description wanted. Instead of just eliminating nogoods from labels computed otherwise, we can conjunctively combine the nodes' labels with the negation of the maximal label for $\perp$. If we have e.g. $\mathsf{maxlabel}(A) = \alpha\beta + \gamma$ and $\mathsf{maxlabel}(\perp) = \gamma\delta + \beta$, we would have to eliminate $\alpha\beta$ to get the maximal consistent label γ for A. Instead we can directly compute only consistent labels if we conjunctively combine a node's label with the negated maximal label for $\perp$. In our case we have $(\alpha\beta + \gamma)\overline{(\gamma\delta + \beta)}$, which can be simplified to $\overline{\beta}\gamma\overline{\delta}$.

This label is in fact a label for A and also consistent. From the view of LL it is of course not the maximal consistent label for A, because we have just shown that this is γ. How then should it be interpreted?

The author favours a generalization of the set representation, interpreting $\overline{\beta}\gamma\overline{\delta}$ as "all the sets containing at least the item represented by γ, but *not* the items represented by β and δ. If we rewrite the label to $\gamma\overline{(\beta + \delta)}$, we note a close connection to Martins and Shapiro's restriction sets (cf. chapter 2): The origin set of node A is described by γ, and the sets characterized by β and δ may not be united with it without getting a contradiction.

This procedure only works for simple sets, as in this case maximal labels are always positive and thus the splitting into origin set and restriction set is possible through the distinction between positive and negative literals, as will be shown below. This is not possible in the general case. For the description of SNeBR this does not matter, because we have only basic sets here: Every hypothesis is labelled with a fresh atomic label.

In the SNeBR system origins sets are always single environments. In our case this corresponds to maximal labels that can be written as simple conjunctions of atoms. Given such a label, we show that the label resulting from conjoining it with the negation of the maximal label for $\perp$ indeed corresponds to the origin/restriction-set representation.

Assume we have a label of the form $\alpha_1 \ldots \alpha_n$, where all the α_i are atomic. Let the maximal label of $\perp$ be $\beta_{11} \ldots \beta_{1n_1} + \cdots + \beta_{m1} \ldots \beta_{mn_m}$. We then get

$$\alpha_1 \ldots \alpha_n \overline{(\beta_{11} \ldots \beta_{1n_1} + \cdots + \beta_{m1} \ldots \beta_{mn_m})} =$$

$$\alpha_1 \ldots \alpha_n (\overline{\beta_{11}} + \cdots + \overline{\beta_{1n_1}}) \cdots (\overline{\beta_{m1}} + \cdots + \overline{\beta_{mn_m}})$$

Multiplying this out and transforming it to DDNF gives either $\perp$ or a label of the form

$$\alpha_1 \ldots \alpha_n \overline{\delta_{11}} \cdots \overline{\delta_{1l_1}} + \cdots + \alpha_1 \ldots \alpha_n \overline{\delta_{k1}} \cdots \overline{\delta_{kl_k}}$$

which then can be retransformed to

$$(\alpha_1 \ldots \alpha_n) \overline{(\delta_{11} \cdots \delta_{1l_1} + \cdots + \delta_{k1} \cdots \delta_{kl_k})}$$

which is of the required form: The first conjunct is the origin set, whereas under the negation bar we find the restriction set, consisting of several sets.

Martins and Shapiro demand that two conditions are met by restriction sets:

1. If α is the origin set and ρ the restriction set, then there is no $r \in \rho$ such that $r \cap \alpha \neq \emptyset$.

2. There are no two different $r \in \rho$ and $s \in \rho$ with $r \subset s$.

It is easy to see that these conditions are satisfied here.

They then describe how origin set and restriction set of a derived item can be obtained, when the respective information is known of the parents. In the cases where the origin set is just computed as set union of the parents' origin sets, the restriction set is computed as

$$\mu(\{r_1, \ldots, r_m\}, \{o_1, \ldots, o_n\}) = \sigma(\Psi(r_1 \cup \cdots \cup r_m, o_1 \cup \cdots \cup o_n))$$

where

$$\Psi(R, O) = \{\alpha \mid (\alpha \in R \wedge \alpha \cap O = \emptyset) \vee \exists \beta \; (\beta \in R \wedge \beta \cap O \neq \emptyset \wedge \alpha = \beta \setminus O)\}$$

and

$$\sigma(R) = \{\alpha \mid \alpha \in R \wedge \neg \exists \beta \; (\beta \neq \alpha \wedge \beta \in R \wedge \beta \subset \alpha)\}.$$

This looks a little complicated, but a closer look reveals that this is exactly what happens if one conjunctively combines labels in the representation presented above.

We now briefly describe how SNeBR can be modelled within our framework. We have already seen how the origin/restriction-set representation of labels can be adopted by our approach.

SNeBR is assumption-based, so we can in principle proceed as in the case of ATMS. The use of relevance logic on first view seems to complicate the matter, but we shall show that we can do without.

The sole purpose of using a relevant implication is that no irrelevant assumptions are recorded. But this is guaranteed as well if we make sure that only maximal labels are computed. Consider again the example from chapter 2: If we have assumptions α:"John is tall" and β:"John is fat", we might of course have a natural deduction style basic logic that allows for AND-introduction and AND-elimination, thus leading to the deduction of $\alpha\beta$:"John is tall and fat" and then $\alpha\beta$:"John is fat". But as we already have β:"John is fat", we know that $\alpha\beta$ is not maximal for "John is fat".

So actually there are no real complications in the modelling of SNeBR, and we can really get a model theoretic characterization without regress to any relevance logic, *provided we have one for the basic logic*, which here is the logic underlying the particular natural deduction calculus they use.

5.4 Combinations: The General Case

We have looked at the most prominent representatives of the main approaches for Reason Maintenance. It should be clear — at least in principle — how a modelling can be done for others that fit into the classification scheme given by the two orthogonal lines justification based vs. assumption based approach (or to be more concrete: single context vs. multiple contexts) and the nonmonotonic vs. monotonic division, the other details being implementation issues in essence.

In order to test the generality of our approach we start with ATMS-like systems. There are several possible generalizations of the ATMS.

1. The basic logic used need not be classical propositional logic. In particular formulae need not be of such restricted forms as being Horn.

   This is already included in our approach. Nothing in our ATMS description depends on the particular choice of basic logic, or a peculiar form of the formulae. So we can in fact freely vary the basic logic.

2. Contradiction detection could be performed by the RMS component.

   This is also included for free, because we do not separate RMS and problem solver. If a problem solver derives the formulae A and $\neg A$, an ATMS generates two new nodes, say B and C, for this and has to be *told* that they contradict one another. However, in our modelling nothing prevents us from directly taking α:A and β:$\neg A$, which enables the RMS to detect $\alpha\beta$ as a nogood without help from outside, because it knows how to deal with the semantic content of the nodes.

3. An ATMS only knows of its nodes and does no inferences on its own. Therefore non-atomic queries can not be performed. If we have e.g. α:A and β:B we can see that in the environment described by $\alpha\beta$ we have $A \wedge B$. Or, if we have have nodes with internal structure, α:$A \vee B$ and β:$\neg A \vee B$ should entail $\alpha\beta$:B. All this must be done by the problem solver and is not detected by the ATMS.

Again, even our modelling of ATMS contains these inference possibilities. The reason is, that theorem 5.1.2 need not be restricted to atomic propositions, but holds for arbitrary formulae, as does theorem 4.1.15.

Within our framework we can also vary the nogood handling:

- We can independently compute maximal labels for nodes and $\perp$, as it is done in de Kleer's ATMS.
- We can use the generalized set representation to keep nogoods in the form of restriction sets like in the Martins and Shapiro system.

Furthermore we are not restricted to always computing maximal labels. We are completely free to choose another type of entailment that allows for the deduction of intermediate labels that are not maximal, but nevertheless sound wrt. the chosen entailment relation. Choices are

- Simple LL-entailment gives us systems not bound to come up with the "best" result, but capable of deriving intermediate results.
- The introduction of the maximality criterion will enforce a behaviour like the one found in most existing systems: The system gives an answer only if it comes to a quiescent optimal state.
- The consistency demand yields the introduction of nogood handling.
- All combinations are possible, as well as using the approximative entailment relations also presented in chapter 4.

Now we have to talk about the introduction of nonmonotonic justifications into ATMS, as well as transition to single contexts.

Single context approaches like TMS differ from ATMS in that for every node the state must be fixed, thus arriving at one consistent state. This criterion is represented by the demand that labels be input complete. Although there can still remain more than one label, there is no need to decide for one particular of these, as this only reflects the nondeterminism present in TMS.

The introduction of nonmonotonic justifications for the general case where arbitrary formulae are admitted as nodes, does not work satisfactorily. This is because in our JTMS representation we have exploited the fact that all nodes are represented by atoms. We could map the non-derivability of a node to the consistency of a label attached to the contrary. If, however, we admit arbitrary formulae instead of just atoms for nodes, we cannot do this any more, because e.g. assuming $A \vee \neg B$ `out` is in fact the same as assuming $\neg A \wedge B$ `in`, which is usually not intended. This is, however, a general problem and has nothing to do with our particular approach.

If we stick to the paradigm that nodes are atomic, as it is done in all existing systems, then there is in fact no problem with introducing nonmonotonic justifications into ATMS, for we can proceed just like in the JTMS case, but simply drop the input completeness demand.

It is also possible to translate a TMS into an ATMS (Giordano & Martelli, 1990*a*), which then performs nonmonotonic reasoning, and then model this ATMS with our approach.

We can conclude that we really have been successful in the primary aim of this thesis: We have provided a *unifying framework* for reason maintenance, which in particular eases the task of finding a model theoretic semantics for the system under consideration. Of course that does not render any of the existent systems useless, as we have not provided a calculus yet. However, we now show how this can be done in principle, although not necessarily very efficiently.

The main advantage of our characterization lies in the fact, that we use the same formalism to describe both justification based and assumption based approaches, thus making it possible to compare the different systems on that basis. Furthermore we are free to change the basic logic (concerning a change in the logic of the labels cf. chapter 8 and the notes in the further works chapter). By changing the basic logic, we can dispose of the tight restrictions of existing systems, like the choice of Horn logic or propositional logics. The first extension has already been made by Reiter and de Kleer in Reiter & de Kleer (1987) and therefore taken into account in works like e.g. Inoue (1990). Arguments for the admittance of full first order logic have been given in the introductory section, so we do not repeat them here.

Chapter 6

Comparison with other Semantics

In this chapter we compare our approach to other semantics that have been proposed for RMSs in the literature.

6.1 ATMS Semantics

The ATMS problem has been shown to be related to *abduction* already by Reiter & de Kleer (1987). Ordinary justifications can be viewed as propositional Horn clauses, nogood justifications as purely negative clauses, whereas the assumptions are always atoms and serve as the *abducibles*. The CMS is a more general form of ATMS, where arbitrary *clauses* are admitted as justifications, and where queries can also be clauses instead of just atoms. If A is the conjunction of all the justification clauses and C the query, then the *answer* of the CMS is the *shortest clause* S with $A \vdash S \vee C$, but not $A \vdash S$. Because of the deduction theorem this is equivalent to $A \wedge \neg S \vdash C$ but not $A \wedge \neg S \vdash \bot$, where $\neg S$ is a conjunction of literals, the *simplest explanation*. In this context, simple means that no proper sub-conjunction $\neg S'$ is an explanation, i.e. $A \wedge S' \vdash C$.

Finding the clauses that correspond to simplest explanations is dual to looking for prime implicants. The dual procedure to the consensus method turns out to be ordinary resolution (Robinson, 1965)[1].

For a further generalization of the CMS Inoue (1990) gives a semantics with a model and proof theory in terms of abduction. Here we have a set W of formulae (not necessarily propositional) as justifications, where assumptions are *ground literals*, and a query G is an arbitrary closed formula.

Then an *explanation* H of G from (W, A) satisfies

- H is a conjunction of literals
- $W \cup \{H\} \models G$

[1] Reiter and de Kleer therefore speak of *prime implicants* in the cited article, though actually *prime implicates* are meant.

- $W \cup \{H\}$ is satisfiable

An explanation is *minimal*, if no sub-conjunction H' of H is an explanation, i.e. $W \cup \{H'\} \models G$.

This can be mapped exactly to our approach, which is in fact a further generalization, because we admit arbitrary formulae for justifications as well as for assumptions and queries. Of course this has an influence on algorithms, but it does not change the abductive semantics.

Levesque (1989) uses a similar approach for examining a more general case of abduction. In this context he finds a particular case corresponding to the ATMS:

> "When belief is closed under logical implication, the corresponding form of abduction is precisely what is performed by the ATMS as characterized by Reiter and de Kleer."
>
> "However else it has been characterized in the past, this theorem establishes that an ATMS can be understood as computing all simplest explanations with respect to this type of implicit belief. Among other things, this guarantees that Poole's account of abduction (with the addition of the notion of simplicity defined here) also specifies the task performed by an ATMS."

The approach of Poole mentioned (Poole, 1988*a*) will be dealt with in the chapter on default handling (chapter 8), as it turns out that his handling of defaults and ours are very close.

Fujiwara & Honiden (1990) describe the (basic) ATMS by a translation to propositional Horn clauses. In contrast to our approach they have an extra handling for contradictions. Their characterization runs via a *smallest fixed points semantics*. ATMS labels represent a subset of a Boolean algebra, a so-called *upper closed set*. "Upper closed" means every superset of a set contained is also contained. This corresponds to our minimal DDNF representation. They also use the notion of a characteristic formula[2] χ, which is the opposite of our characteristic function. This is unique up to logical equivalence. An ATMS is a quadruple (N, A, J, C) like defined here, A being a subset of N, J a subset of $N \times 2^N$, C a subset of J (the nogood justifications). Their upper closed sets $l(J, n)$ and $i(J, C)$ are such that

$$l(J, n) = \mathsf{maxlabel}(n)$$

and

$$i(J, C) = \mathsf{maxlabel}(\perp).$$

[2] *Characteristic clause* first appears in (Bossu & Siegel, 1985) in this context:

> "Informally speaking, characteristic clauses are intended to represent 'interesting' clauses to solve a certain problem, and are constructed over a sub-vocabulary of the representation language." (This is called a *production field* in (Inoue, 1990)).

The computed $l(J,n) \setminus i(J,C)$ is not necessarily upper closed, so the upper closure (ucl) has to be taken. A label then is the prime implicant decomposition of $\chi(\mathsf{ucl}(l \setminus i))$.

Despite the fact that (Fujiwara & Honiden, 1990) certainly fulfills the task of proving the ATMS labelling procedure correct, their semantics is doomed to be of minor importance, since the semantical characterizations of assumption based methods in terms of abduction, as been presented above, include much broader classes than the basic ATMS.

Summing up we can say that concerning the ATMS our approach is very closely related to some known proposals. With the exception of the last mentioned, specialized approach, the ATMS characterizations use well-known semantics and can be generalized in several ways very easily. So the problem of ATMS semantics can be viewed as solved. Nevertheless none of the authors mentioned *how general* the abductive approach really is. In particular, except for (Poole, 1988*a*) all former approaches are limited to propositional logics (this certainly comes from the fact that this was not seen as necessary, because of the two component approach). Furthermore, we have shown that it is possible to describe justification based systems like the JTMS using the same formalism (cf. chapter 5).

6.2 Semantics for Justification Based Approaches

Semantics of JTMS have to deal with nonmonotonic justifications. In addition, the demand of well-foundedness proved to present some problems. There are a lot of formalisms proposed for JTMS that suffer from the fact that they are only developed for that very reason, e.g. (Brown, 1985; Brown, Gaucas & Benanav, 1987; Brown & Shoham, 1988; Morris, 1988*c*; Brown, 1988), and therefore do not allow easy comparison of the characterized systems to others.

There is a paper by Haneclou (1987) that characterizes nonmonotonic logics (including reason maintenance) via *valuations* (the truth values `true`, `false`, `undefined` and `contradiction` form a lattice), which is rather general, especially independent of any concrete implementation and control strategy, but the results cannot be directly used for our purposes.

Direct characterizations for JTMS are e.g. (Junker & Konolige, 1990*b*) or (Xianchang & Huowang, 1991). The former does not deal with CP-justifications, whereas the latter shows that they can in fact be eliminated. Junker & Konolige (1990*b*) do not want to characterize TMS in the first place, but use it to compute extensions of autoepistemic and (Reiter's) default logic, but indirectly this results in a characterization of TMS, too. They use a semantics developed by Reinfrank, Dressler & Brewka (1989). These map TMS justifications to formulae of autoepistemic logics. The justification with OUT-SET $\{B\}$, IN-SET $\{A\}$ and consequent C becomes $La \wedge \neg Lb \rightarrow c$, where L is the AEL modal operator with La meaning "a is believed". The TMS is then proven to compute all *non modal* atoms of a *strongly grounded* AEL-extension. Reinfrank, Dressler and Brewka explicitly mention the limits of their "NMFS-theory" (non monotonic formal system): It does not work even for a full propositional language, but is dependent on the formulae being Horn.

Xianchang and Huowang claim, that their "semantics" is equivalent to that of Junker and Konolige, but no proof is given.

A major part of the JTMS characterizations use some kind of *stable model semantics*. This is not astonishing, since the groundedness condition (which corresponds to well-foundedness) directly relates to *negation as failure*. Stable models appear in logic programming, where they have been introduced by Gelfond & Lifschitz (1988). They are used for the characterization of nonmonotonic RMS e.g. by Elkan (1990) (or already Elkan (1989)). Elkan shows that his models are grounded iff they are stable. Finding a stable model is proven to be NP-hard (Elkan, 1989). A JTMS may possess none or more than one.

> "One can view the nodes of a nonmonotonic TMS as propositional calculus atoms, and its justifications as implications of a restricted nature. A set of believed nodes corresponds to a propositional calculus interpretation."
>
> "A justification is a *directed* propositional clause."

It is noted that justifications are not correctly modelled if translated to (material) implications, but the use of contrapositives has to be blocked, because justifications are to be understood as *directed*.

Another stable models characterization is due to Pimentel & Cuadrado (1989).

The stable models semantics is not able to capture dependency directed backtracking. There are several papers from Giordano and Martelli on that theme. Giordano & Martelli (1990*b*) introduces *generalized stable models*, which facilitates semantics for JTMS including the dependency directed backtracking procedure. It is noted that in spite of the directed nature of justifications the use of *contrapositives* is necessary for resolving inconsistencies.

The fact that not all contrapositives are allowed, for it does not suffice that the consequent is `out` by default assumption, but its state of being `out` has to be *proved*, led to the suggestion to use 3-valued logics (with truth values `in`, `out` and `false`), as e.g. in (Giordano & Martelli, 1990*d*). Eshghi & Kowalski (1989) show that it is possible to translate such a three valued net into an ordinary two valued one by adding further justifications which simulate the backward rule applications. There is a direct correspondence between the "use all contrapositives in a three valued logics" and the "restricted use of contrapositives in a two valued setting" approaches.

The fact that dependency directed backtracking actually modifies the set of justifications (cf. chapter 2) led to the omission of DDB in many models. The stable closure solution in AEL (Morris, 1988*b*) does in a sense something similar to DDB, but yields unexpected results from the standpoint of reason maintenance.

A mixed approach (3-valued stable models) called *skeptical model* is given by Witteveen (1990). The skeptical model is $O(n^2)$-computable. An interesting remark in (Witteveen, 1990) is

> "The solution to these problems can be found if we are prepared to give up the idea of a *complete* evaluation of beliefs."

which is exactly what we propose.

Various authors try to give a JTMS semantics in terms of AEL (Moore, 1985), so e.g. Reinfrank et al. (1989) or the already cited Elkan (1990). Reinfrank et al. (1989) give as TMS characteristics

- finiteness (logical incompleteness): this is accounted for in our approach by the input completeness criterion
- the nondeterministic choice of an extension (so we give all of them)
- global groundedness
- asymmetric use of justifications

Their method is based on (Konolige, 1988), who shows "equivalence" between DL and AEL (kernels of strongly grounded AEL extensions are proven to be DL extensions). DDB is detected as a problem by Reinfrank et al., but not dealt with. Therefore they omit nogood inferences and backtracking, because that could possibly destroy groundedness.

The same holds for (Fujiwara & Honiden, 1989).

Our approach differs very much from all these proposals. We have shown that it is possible to use an *assumption-based* framework for justification-based systems as well. This is done by introducing as assumption the hypothesis which is implicit there, namely that everything can be assumed `out` without further reason. Then we of course get multiple contexts. The transition to single contexts is *not* modelled, because the particular choice of context is done nondeterministically anyway[3]. However, the exclusion of not well-founded solutions is reflected. We do this by forcing not well-founded environments to become inconsistent. The same is done when we handle nogoods. Here we follow the spirit of Elkan (1990). This is why we come to a correct nogood strategy.

6.3 Unifying Frameworks

There are in fact none. Because of their apparent differences, assumption-based and justification-based approaches have always been dealt with separately, when concerning semantics. The attempt of McDermott, who at least spotted the problem, has already been discussed and recognized as insufficient (cf. section 3.2). This insufficiency has been the major motivation for writing this thesis.

However, there are numerous papers in literature that show the close correspondences between single pairs of approaches. Examples are:

[3]Of course it is actually deterministic in real systems, but the particular choice is intendedly not part of their semantics.

- ATMS and Constraint Solving: the claim is made that ATMS be more general (de Kleer, 1989)
- TMS as implementation of default logic (Junker & Konolige, 1990*b*)
- TMS, stable models and AEL (Elkan, 1990)
- ATMS and abduction (Inoue, 1990; Poole, 1988*a*)
- AEL and DL (Marek & Truszczyński, 1989)
- AEL and circumscription (Konolige, 1989)

What DL and JTMS have in common is that one has to decide for one extension at a time. The extension chosen, however, is not *semantically* distinguishable from others (Hanks & McDermott, 1986).

The strong connection between all these approaches, as it is suggested by the numerous pairwise correspondences, is made explicit by our approach, which supplies a uniform presentation, including the close relationship to default reasoning, as shown in chapter 8.

Chapter 7

Incremental Calculi

So far we have been concerned with theoretical, i.e. semantical aspects. The result was a uniform formalism for the description of Reason Maintenance Systems. This can now be used to build practical inference systems directly operating with labelled logic.

It will turn out that such systems can be constructed by minor modifications of calculi for the respective basic logics — in a canonical manner. This leads to the advantageous situation that existing provers for the basic logic can be modified to do their own reason maintenance. Instead of giving the problem solver an RMS as an assistant (and getting all the problems with the interface described in chapter 3), the problem solver does the maintenance of dependencies all by itself.

It has to be admitted that the proposed canonical transformation in its original (rather naïve) form is too simple to result in systems that expectedly treat reason maintenance in an *efficient* way. Therefore we shall certainly not gain anything in the case of classical propositional logic as basic logic. Quite the opposite is true: If we attempted to reimplement de Kleer's ATMS this way, for instance, we were confronted with a horrible loss of performance.

Our approach will however turn out to be interesting if the basic logic is not decidable. Every trial to built a system like ATMS, that in a single "run" computes the maximal labels for all the nodes, is doomed to fail. In these cases systems like the ones proposed here may not be the most efficient solution, but they are a solution at all. Concerning efficiency there are some hints in the section on proof strategies (7.2).

We call our systems *"incremental systems"*. The name captures the fact that maximal labels are approximated step by step. Their advantage is, that a procedure which else could only be done with global consideration of the whole set of input formulae (cf. the computations done by e.g. ATMS or TMS) can in fact be attacked by purely *local* inference rules. All the intermediate "lemmata" may of course not meet the maximality criterion, but are – with respect to our well-defined semantics of labelled logics (cf. definition 4.1.7) valid theorems. That means procedures implementing such an incremental calculus have the status of an *anytime procedure* (Dean & Boddy, 1988; Haddawy & Frisch, 1991), i.e. intermediate results are meaningful, and the result only improves, if the algorithm is given more time.

If the basic logic is semi-decidable and the chosen BL-calculus (refutation–) complete we can also prove some kind of completeness for the transformed calculus. Given a (refutation–) complete calculus $\mathcal{C}$ for BL, we transform it into a canonical calculus $\mathcal{C}'$ for the labelled logic with BL as basic logic (we denote this by LL(BL)). If we are interested in the maximal label for a BL-formula F we imagine just to give F as a query and then get $\alpha = \mathsf{maxlabel}(F, \Phi)$. But the completeness of $\mathcal{C}'$ only guarantees that $\alpha{:}F$ will be derived some time. It also guarantees that a potential query of $\alpha{:}F$ will certainly be answered in the affirmative. The completeness does *not at all* mean that the calculus will identify α as being the maximal label for F. A query like "give me the maximal label for F" will thus at best (and this is exactly what we propose) result in a strictly monotonous sequence of labels for F, which approximates the maximal label and is even guaranteed to reach it, but one cannot expect (at least not in general) the algorithm to terminate. Instead it would run on, of course never "overbidding" the maximal label. Of course it may be possible to find termination conditions for restricted classes of formulae.

This sounds rather negative, but one has to be aware of the fact that in the case of an undecidable basic logic one cannot seriously expect anything better. This is immediately reasonable if one considers that the status of maximality of a label for a formula F corresponds to the *non-derivability* of F in other subsets of the set of input formulae.

The situation is even worse if the transition from LL-derivability to *consistent* LL-derivability is done. We can give a local transformed calculus $\mathcal{C}''$ for this case, too. But the lemmata generated by this calculus are such that we now even lose *correctness*. That originates from the fact that at a given point in time also the maximal label for $\perp$ may not be found yet. For this reason the calculus may produce labels for F that are not only not maximal, but maybe even inconsistent. It has to be noted, however, that the loss of correctness only concerns the semantics of $\models_{\mathrm{consLL}}$. With respect to (increasing) chains of $\models_{\omega\text{-LL}}$ semantics the derived formulae can be proved correct.

In the sequel we first present the basic idea of how to transform a calculus for BL into one for LL(BL). After this we give the correctness and completeness results of interest. Some considerations and hints on how to improve a real system follow. For the whole chapter we assume BL is an arbitrary, but monotonic, logic.

7.1 From a BL-Calculus to its Corresponding LL(BL)-Calculus

From chapter 4 we know that (the DDNF of) a positive, relevant label represents subsets of the set of input formulae. If this set is basic, every atomic label directly corresponds to exactly one input formula and the correspondence can easily be seen. From this it is rather straightforward to interpret a formula

$$\alpha_{11} \cdots \alpha_{1n_1} + \cdots + \alpha_{m1} \cdots \alpha_{mn_m}{:}F$$

as "F is derivable from the set $\chi(\alpha_{11} \cdots \alpha_{1n_1})$ as well as from the set ... as well as from the set $\chi(\alpha_{m1} \cdots \alpha_{mn_m})$", where we could dispose of the χ, because $\chi(\alpha)$ represents a one element set, if α is atomic and relevant.

Definition 7.1.1 (Labelled Representation, Labelled Calculus)
Let Φ be a set of BL-formulae and $\mathcal{C}$ a calculus for BL with rules of the form

$$\frac{F_1, \ldots, F_n}{G}. \tag{R$_i$}$$

Then the **labelled representation** of Φ is a set LL(Φ) of LL(BL)-formulae obtained from Φ by prefixing each formula in Φ with a "fresh" atomic label.

The **labelled calculus** $\mathcal{C}'$ to $\mathcal{C}$, working on LL(BL)-formulae, is obtained from $\mathcal{C}$ by transforming each rule R_i into its labelled version

$$\frac{\alpha_1{:}F_1, \ldots, \alpha_n{:}F_n}{\alpha_1 \cdots \alpha_n{:}G}, \tag{R$'_i$}$$

the α_i being label variables, and adding a further rule

$$\frac{\alpha_1{:}F, \ldots, \alpha_n{:}F}{\alpha_1 + \cdots + \alpha_n{:}F}, \tag{CR}$$

the so-called **contraction rule**.

Remark 7.1.2
For any set Φ of labelled formulae, LL(Φ) *is basic.*

Remark 7.1.3
Given a basic set (such as the labelled representation of a set of BL-formulae), in $\mathcal{C}'$ only formulae with positive, relevant labels can be deduced.

This immediately shows, that $\mathcal{C}'$ can not be complete wrt. the semantics of $\models_{\mathrm{LL(BL)}}$. However, we can show some important properties.

Theorem 7.1.4 (Properties of the Labelled Calculus)
The labelled calculus has the following properties:

1. *Every rule R_i, which is correct wrt. the semantics of BL, will result in a rule R'_i, which is correct wrt. the semantics of LL(BL).*
2. *CR is correct wrt. the semantics of LL(BL).*
3. *If $\mathcal{C}$ is (refutation-) complete wrt. the semantics of BL, then $\mathcal{C}'$ is (refutation-) complete wrt. the semantics of LL(BL) for* maximal *formulae.*

Before we give the proof, let us make some comments on the third assertion:

A simple example may show, why for arbitrary sets of formulae $\mathcal{C}$ can not be complete even for positive relevant labelled formulae: Let $\Phi = \{\alpha + \beta{:}F\}$. The construction of the rules R_i resp. CR is such that labels can only be "constructed", not "decomposed". That means that e.g. $\alpha{:}F$, which of course is LL-implied, is not derivable in $\mathcal{C}'$.

Now one could object that the example is ill suited, since our transformation always generates basic sets of formulae. As for basic sets of formulae $\models_{\mathrm{LL}}$ and $\models_{\mathrm{equivLL}}$ coincide, the calculus is complete for positive relevant formulae.

This is correct, but we shall need the weaker version stated here later on. When we talk about proof strategies, there will be things like the deletion of subsumed clauses or other operations that depend upon the notion of transformations of sets of formulae preserving satisfiability. These transformations will in general not leave basic sets basic.

Example 7.1.5
The set of formulae $\{\alpha{:}F, \beta{:}F\}$ is basic. Application of CR gives $\alpha + \beta{:}F$. This subsumes $\alpha{:}F$ as well as $\beta{:}F$. We therefore transform the original set to $\{\alpha + \beta{:}F\}$[1] ending up with exactly the problem described above. The same will already happen if a transformation to clause normal form is attempted. Because of duplication of labels ($\alpha{:}F \wedge G$ will become $\alpha{:}F$, $\alpha{:}G$), the resulting set of labelled clauses is not basic.

Now we shall prove theorem 7.1.4
Proof:

1.) Correctness wrt. the semantics of BL means

$$\{F_1, \ldots, F_{\mathrm{n}}\} \models_{\mathrm{BL}} G.$$

Now let $\alpha_1, \ldots, \alpha_{\mathrm{n}}$ be arbitrary labels. We have to show

$$\{\alpha_1{:}F_1, \ldots, \alpha_n{:}F_n\} \models_{\mathrm{LL(BL)}} \alpha_1 \cdots \alpha_n{:}G.$$

Let us call the set $\{\alpha_1{:}F_1, \ldots, \alpha_n{:}F_n\}$ Φ. There is a subset of Φ, namely Φ itself, for which, according to our assumptions, we have $\mathsf{formula}(\Phi) \models_{\mathrm{BL}} G$. Further $\mathsf{label}(\Phi)$ equals $\alpha_1 \cdots \alpha_n$, so the second condition of definition 4.1.7 is satisfied trivially.

2.) To show: $\Phi = \{\alpha_1{:}F_1, \ldots, \alpha_n{:}F_n\} \models_{\mathrm{LL(BL)}} \alpha_1 + \cdots + \alpha_n{:}F$. Take all one element subsets of Φ. Certainly $F \models_{\mathrm{BL}} F$ holds. The rest is trivial.

3.) Let $\mathcal{C}$ be complete and Φ the given set of axioms. Furthermore

$$\Phi \models_{\max(\mathrm{LL})} \alpha{:}F$$

holds. Now consider $\mathsf{formula}(\Phi)$. Let $\Psi_1, \ldots, \Psi_{\mathrm{n}}$ be all minimal sets (wrt. set inclusion) for which $\mathsf{formula}(\Psi_i) \models_{\mathrm{BL}} F$ holds. $\mathcal{C}$ being complete, there are derivations of F by applications of $\mathcal{C}$-rules in all the Ψ_i. One can simply "lift" the derivations to $\mathcal{C}'$ and finally apply CR to get a proof for $\beta{:}F$ with $\beta = \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$. It remains to be shown that $\alpha \leftrightarrow \beta$.

[1] If it seems to be suspicious how one could assume those sets as equivalent, then note that the equivalence is only given concerning $\models_{\mathrm{LL}}$, not concerning $\models_{\mathrm{equivLL}}$.

This is simple: Since $\alpha{:}F$ is an LL-consequence of Φ, there are sets $\Theta_1, \ldots, \Theta_{\mathrm{m}}$ with $\mathsf{formula}(\Theta_i) \models_{\mathrm{BL}} F$. Since the Ψ_i above are minimal, all the Θ_j are supersets of some Ψ_i. From this it follows that $\bigvee_i \mathsf{label}(\Psi_i) \rightarrow \bigvee_j \mathsf{label}(\Theta_j)$, from which the desired equivalence is obvious because of the maximality of α.

The corresponding result for refutation completeness directly follows, if the definition of a refutation proof is considered. □

As a further result we can get

Proposition 7.1.6 (Decidability Results)
If BL is semi-decidable, so is LL(BL). If BL is decidable, then LL(BL) is decidable, too.

This is immediately obvious. However, even in the decidable case the complexity of the task is higher. This seems plausible, for a proof of $\mathsf{maxlabel}(F){:}F$ has to take into consideration *all* "minimal" proofs of F. In the decidable case also the question "what is the maximal label for F?" is decidable. A naïve decision procedure is, e.g. to answer the question $\Psi \models_{\mathrm{BL}} F$? for all subsets Ψ of Φ. Certainly one could save some of these tests, since, if the answer is "no" for some Ψ, the same answer holds for all subsets of Ψ, and if the answer is "yes", this is also true of all supersets (remember BL is monotonic).

Now we come to the question whether we could obtain transformed calculi also for the case of consistent consequence. The answer is yes, if we are content with a calculus that produces incorrect lemmata. What we aim at is to generate derivations for the (BL-)theorem F in question as well as for falsity, thus looking for nogoods. As well as we are never sure if we have arrived at the maximal derivation for F, we are not able to tell whether the nogood found so far is maximal. We propose to *filter* the derived label of F through the nogoods known to "sieve them out". As we possibly do not know the maximal nogood, the result cannot be guaranteed to be correct wrt. the semantics of $\models_{\mathrm{consLL}}$. It is, however, correct wrt. $\models_{\omega\text{-LL}}$ semantics with ω being the known nogood. As this known nogood monotonically increases over time and will not only approach, but definitely reach $\mathsf{maxlabel}(\bot)$, the results will be correct wrt. increasing $\models_{\omega\text{-LL}}$ semantics, and thus eventually even wrt. the semantics of $\models_{\mathrm{consLL}}$. The details will be elaborated in section 7.3.

7.2 Proof Strategies

As already noted, the naïve calculi obtained via the transformations given in the preceding section will not be of much use in practice. Especially if our aim is to use a slightly modified version of an existing proof system for BL, we shall soon notice that the simple equation PROOF-SYSTEM = AXIOMS + RULES does not suffice. In reality often the use of *proof strategies* is of great importance for the performance of a system. So we should dedicate some thoughts to the examination of the question, which proof strategies can be taken over into labelled systems.

We shall not answer this question exhaustively, but simply analyze some important examples. For doing so, we take BL to be first order predicate calculus and think of some resolution based refutational proof system.

7.2.1 Transformation to Clause Normal Form

In section 4.3 we have learned that equivalence preserving transformations within the formula part of labelled formulae are allowed. If we are only looking for satisfiability, as is done within refutationally working systems, usually skolemization is performed, which does not preserve validity, but keeps satisfiability. It is not difficult to see, that steps of this kind will not do any harm. Within CNF-transformation, the most critical point from our view is the splitting of conjuncts into different clauses. But, as can be seen by theorem 4.3.11, even this works. We have to note, however, that it could undermine a formula set's status of being basic.

7.2.2 Set of Support (SOS)

One of the most common strategies for diminishing the search space in automated theorem proving is the set of support strategy. Its applicability depends heavily on the characteristic setting theorem provers are normally used in, i.e. mathematics. There we have the situation that the axioms can be assumed to be *consistent*. This means that a proof of falsity must necessarily contain the negated theorem, a fact which can be exploited during the search. In our setting this is simply not true in most cases, as the set of axioms will often be (BL-)contradictory. So SOS will not be of great help here in general. Whether it can nevertheless be exploited, depends on the particular interest. If one is really concerned about computing LL-theorems or wants to track the effects of inconsistencies in the original formula set, using SOS produces wrong answers. If, however, the goal is to obtain *maximal consistent* labels for BL-formulae, as in the abstract reasoner approach we sketch in section 7.5, SOS can be used, if the process is divided into finding the maximal label for $\perp$ and finding the maximal label for the formula in question, with the intention to "subtract" the former from the latter. In this configuration one can use SOS for the second process, which then does not compute the true maximal label, but the part of the label that is missing would be deleted afterwards anyway.

7.2.3 Deletion Strategies

7.2.3.1 Tautologies, Purity etc.

The idea behind these strategies is not to examine clauses that simply cannot be part of a proof. As the definition of labelled consequence is based on BL-consequence, it can easily be seen, that deletions of this kind can be used the same way in LL(BL) as in BL.

7.2.3.2 Subsumption

The case with subsumption is not that easy as are the deletions mentioned in the preceding paragraph. The difference is, deleted subsumed clauses are not guaranteed not to be part of a proof, but their deletion is justified by the fact, that, if they are, there is still another proof possible using the subsuming clause instead. If we are interested in "better" (ideally maximal) labels we have to look for every possible derivation, however, so we cannot dispose of proofs that easy. If we are, e.g. given $\{\alpha{:}A \wedge B, \beta{:}A\}$, we immediately notice that (regarding only the formula parts) the first formula subsumes the second one. If we deleted that formula, we are no more able to derive $\alpha + \beta{:}A$, which is maximal.

Nevertheless there is a possibility of subsumption deletion even in the labelled calculus, if we are only interested in formulae with maximal labels. The solution is, that we have to consider also the labels in the definition of subsumption:

Definition 7.2.1 (Labelled Subsumption)
Given two labelled formulae $\alpha{:}F$ and $\beta{:}G$, we say $\alpha{:}F$ **subsumes** $\beta{:}G$, iff F subsumes G according to the underlying basic logic and further α subsumes β in the logic of the labels.

We then get

Lemma 7.2.2 (Subsumption Deletion)
If the calculus $\mathcal{C}$ for BL allows for subsumption, then the deletion of (LL)-subsumed formulae does not destroy the derivability of any maximally labelled formula.

Proof:
Let $\Phi \models_{\mathrm{LL}} \alpha{:}F$ with $\alpha = \mathsf{maxlabel}(\Phi, F)$. Suppose there is a proof of $\alpha{:}F$ in $\mathcal{C}''$ using some formula $\beta{:}G$ which is LL-subsumed by some $\beta'{:}G'$. This means in particular that G is BL-subsumed by G'. Because $\mathcal{C}$ allows for subsumption deletion, there is another proof in $\mathcal{C}$ using G' instead of G. Because of the construction of LL(BL) we can lift this proof to $\mathcal{C}''$, getting a proof of some $\alpha'{:}F$. The construction also tells that $\alpha' \geq \alpha$. Since α is maximal, this must be $\alpha' = \alpha$. □

7.2.4 Restrictions on the Choice of Inference Rule Applied

In general completeness can only be guaranteed if the strategy of rule application is *fair*, which means that every rule has its fair chance, i.e. will finally be applied, if applicable at all. There are strategies that give preference to some rules over others. Sometimes this does not contradict fairness, because the rules not considered under the strategy can be shown not to be applicable in the respective situation anyway. Sometimes other considerations may justify the selection.

We give here only one example of a rather primitive strategy possible for incremental systems of labelled logic. The strategy can simply be stated as "give the contraction rule precedence over the other rules, i.e. apply it immediately, if applicable".

We show that this strategy is not only fair, but can in fact make the search tree smaller.

Proposition 7.2.3
After application of the contraction rule the antecedent formulae can immediately be disposed of, because they are subsumed by the consequent. Furthermore, if we do so, the strategy of giving CR preference over every other rule is fair.

Proof:
Because the formula parts of all the antecedents as well as the consequent are identical, BL-subsumption is trivial. The construction of the consequent's label as disjunction of the antecedents' labels guarantees the subsumption relation for the label, too.

If we delete the antecedents, the rule application will decrease the number of formulae. So there are no infinite chains of CR-applications. So finally another rule will be chosen. Lemma 7.2.2 guarantees that all rules applicable before will still be applicable in the new situation. □

7.3 A Strategy for Consistent Consequence

At the end of section 7.1 we already talked about the problem to find a calculus for computing consLL-derivatives, given a calculus for the underlying basic logic. Of course the labelled calculus $\mathcal{C}'$ is complete also wrt. the semantics of $\models_{\mathrm{consLL(BL)}}$, for the theorems of $\models_{\mathrm{consLL(BL)}}$ are a subset of those of $\models_{\mathrm{LL(BL)}}$. But such a calculus unfortunately yields incorrect formulae, too.

In order to fix these shortcomings we explain here a proof strategy, which guarantees that the formulae derived will be correct wrt. the semantics of $\models_{\omega\text{-LL(BL)}}$ for monotonically increasing ω.

The strategy is based on directly filtering out all known nogoods from labels the very moment they are computed. We formulate the filtering procedure as another rule, called **nogood filtration rule**,

$$\frac{\alpha + \beta\gamma{:}F, \beta + \delta{:}\bot}{\alpha{:}F}, \qquad \text{NF}$$

but in order to obtain the desired results we have to use a particular strategy for rule application. This becomes clear, if we notice that NF works into a direction opposite to subsumption. The strategy is simply demand that the NF-rule precedence is not only given over all others, but has to be applied after each other rule application before "looking" at their results.

Definition 7.3.1 (Quiescent State)
Let $\mathcal{C}'$ be the labelled calculus for some BL-calculus $\mathcal{C}$. A proof system using $\mathcal{C}'$ as well as subsumption deletion and the strategy of giving CR precedence over all the other rules. Such a proof system is said to be in a **quiescent state**, iff CR is not applicable on the current set of formulae.

Lemma 7.3.2
If a labelled proof system is in a quiescent state, then every formula part occurs at most once in the set of current formulae.

Proof:
If this were not the case, then CR would be applicable. □

Definition 7.3.3 (Labelled Calculus for Consistent Consequence)
The calculus $\mathcal{C}''$ for consistent consequence is obtained from $\mathcal{C}'$ by superimposing the following strategy:

After every application of a rule R_i, CR must be applied until the system is in a quiescent state. Then NF is applied for every formulae with formula part F different from $\perp$, if applicable. The resulting consequence is not simply added to the current set of formulae, but *replaces* the first antecedent. Only after this potential subsumption deletions may be performed. We refer to this whole process as one **derivation step**.

Theorem 7.3.4 (Correctness wrt. ω-LL(BL))
After every derivation step in $\mathcal{C}''$ we have the following situation: Let ω be the (unique) label of $\perp$. Then every formula in the current set of formulae is $\models_{\omega\text{-LL(BL)}}$-entailed. This means that $\mathcal{C}''$ is correct wrt. the semantics of $\models_{\omega\text{-LL(BL)}}$.

Proof:
Let Φ be the set of original formulae. Since $\mathcal{C}'$ is correct wrt. $\models_{\text{LL(BL)}}$, this also holds for $\mathcal{C}''$, which produces only a subset of the formulae derivable by $\mathcal{C}'$. Therefore for any $\alpha{:}F$ in the current set of formulae, we have $\Phi \models_{\text{LL}} \alpha{:}F$.

Now suppose there is a prime implicand β of α with $\beta \to \omega$. Since all the labels are positive, there must be a prime implicand γ of ω, such that the NF-rule is applicable. This cannot be the case.

Because of the transitivity of $\models_{\text{LL}}$ this result can by induction be generalized to an arbitrary number of derivation steps. □

Theorem 7.3.5 (Completeness for Maximal Labels wrt. consLL(BL))
If the underlying calculus $\mathcal{C}$ is (refutation–) complete for BL, then $\mathcal{C}''$ is (refutation–) complete for maximally labelled LL(BL)-formulae with respect to the semantics of $\models_{\text{consLL}}$.

Proof:
As $\mathcal{C}'$ is complete wrt. $\models_{\text{LL}}$ (point 3 in theorem 7.1.4), it is also wrt. $\models_{\text{consLL}}$, for its theorems are a subset. It remains to show that the application of NF does not destroy the derivation of formulae maximally labelled according to $\models_{\text{consLL}}$. This is rather simple: NF deletes formulae of the form $\alpha + \beta\gamma{:}F$, if $\Phi \models_{\text{LL}} \beta + \delta{:}\perp$. It is immediately obvious that $\Phi \not\models_{\text{consLL}} \alpha + \beta\gamma{:}F$ and the same holds for all formulae derivable from this by $\mathcal{C}'$.

It may, however, be the case that the formulae now being deleted have formerly been used to eliminate subsumed formulae. All of those are, however, subsumed by either the newly introduced formula $\alpha{:}F$ or by $\beta\gamma{:}F$ (α as well as

$\beta\gamma$ is positive and therefore for any δ we have that $\alpha + \beta\gamma \to \delta$ implies $\alpha \to \delta$ or $\beta\gamma \to \delta$). Derivatives of the latter are not consistent. $\square$

Corollary 7.3.6
Let ω be the label for $\perp$, concerning the current set of formulae. $\mathcal{C}''$ is complete wrt. the semantics of $\models_{\text{consLL(BL)}}$ and correct wrt. the semantics of $\models_{\omega\text{-LL(BL)}}$, thus approaching correctness wrt. the semantics of $\models_{\text{consLL(BL)}}$ with increasing ω.

7.4 Related Work

Thoughts on incremental systems can be found in (Cadoli & Schaerf, 1992)[2]. There also an approximizing semantics is introduced[3].

> "We are trying instead to formalize the approximation of a logical consequence relation defined by means of an extensional semantics but too difficult to compute, through different, but simpler, consequence relations, also defined by means of an extensional semantics."
>
> "any intermediate step (level of approximation) provides clear information which is semantically related to the final answer; the intermediate steps can be efficiently computed; subsequent steps are computed using information obtained in previous ones."
>
> "In our method the answer to a query is reached — although in exponential time — through the computation of several simple steps."

Complexity results of similar problems are e.g. in (Kautz & Selman, 1991; Eiter & Gottlob, 1991; Gottlob, 1991; Stillman, 1990).

7.5 The Abstract Reasoner: A Visionary View of a System Using Incremental Calculi

In this section we describe in a very general form how the type AI system sketched in the introduction can be built using labelled logics. We only need a rather simple form of labelled logics for this: Semi-basic sets of labelled formulae will be sufficient.

Imagine a system that models an agent's memory. It will be supplied with information from outside, draw inferences autonomously, and from time to time be queried whether some statement holds or not. This query will generally be time critical. The system can not wait for a prover to try an exhaustive search

[2]Or better the original long form (Cadoli & Schaerf, 1991).
[3]The quotations following are from (Cadoli & Schaerf, n.d.).

(possibly infinitely), but has to come up with an answer eventually. What we are arguing for is a system that then delivers *the best answer found so far*, that means an answer based on the *greatest label* found for that query, including the unsatisfactory $\perp$ (note that this would assume looking for F and $\neg F$ in parallel, for the system could be able to show both of them, given inconsistent prerequisites).

As long as there is no query, the system could try to prove $\perp$ from the original set, thus detecting nogoods. It then could perhaps inform the outer world of these nogoods, in order to allow withdrawals.

Take a look at figure 7.1.

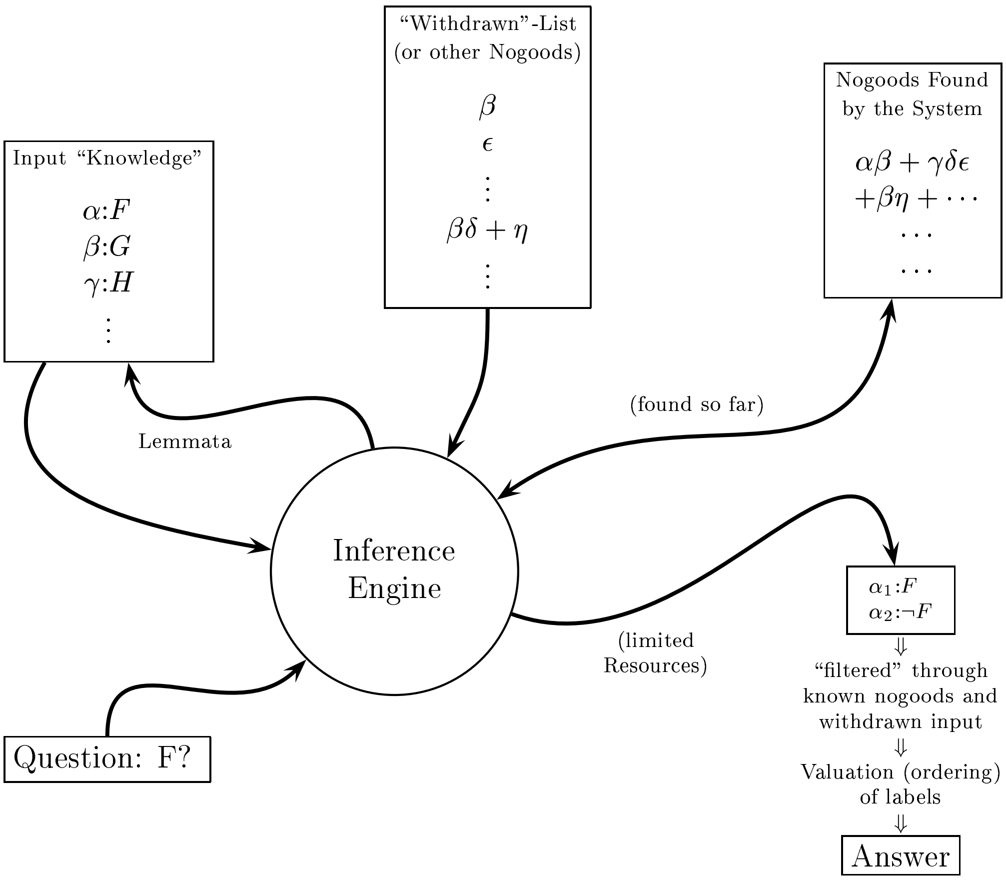


Figure 7.1: the Abstract Reasoner Architecture in closer detail

Every statement given to the AR component is formulated in some basic logic BL. When entered, it gets "stamped" with a fresh atomic label and is included into the knowledge base, which is in principle monotonically increasing.

If an arbitrary question F is asked, the system has to do several things:

1. first of all it tries to derive the item in question. This is done according to an arbitrary procedure suitable for BL. E.g. a refutation proof is perfectly fine. The decisive difference to ATP is, that the procedure should not stop

after having found a derivation, but should go on and look for alternative proofs. This may not end for undecidable BL, but the "raw answer" can only be improved by yielding bigger and bigger labels (wrt. our order given for labels). Of course there may be criteria that reveal a label as maximal. In such cases one could simply stop.

2. if BL is not decidable, even the search for only one derivation may not terminate. As the yes/no question in fact means "can you find a derivation of F or $\neg F$", a second procedure trying to derive $\neg F$ is started in parallel. Note that the whole procedure must not be aborted if one of the two parts yields a derivation, because there may be possibilities to derive both, as the formula set is not guaranteed to be BL-consistent.

3. this procedure is not really necessary, but supplies the system with a better performance. As contradictions may be present in the original formula set already, the AR does not need to wait for a query at all, but can try to look for all refutations of the original set in advance (or whenever there is time for that). Every label found for $\perp$ (BL), can be stored in a separate *nogood database* for future use.

This procedure can be run for an arbitrary time and interrupted at any time. The resulting raw answer may of course not be the correct one, but it can serve as an approximation and is guaranteed to approach the correct solution in a sense described below.

There are now several possibilities, what the concrete answer given by the system could look like. Say $\alpha_1{:}F$ as well as $\alpha_2{:}\neg F$ have been derived within the time available. One solution is to simply output these. One can, however, do better. First of all, every nogood found so far can be eliminated from the raw labels[4]. This is best described as a *filtering* process. If there remains a label only for F *or* $\neg F$, one could translate this as "yes" respective "no".

It has to be noted that this must be handled with care. The yes/no answers are only *temporary* approximations. With increasing time there can occur arbitrarily many changes from yes to no and vice versa. This is due to the fact that, though clearly the labels increase monotonically, the same holds for the nogood labels as well. So the approximation process does not approach the final answer "from below" or "from above", but oscillates.

It may also happen that there does not remain a label for F as well as for $\neg F$. This simply means that there has not been sufficient time to find a (consistent) proof at all. Quite the opposite can occur also. There may be (different) labels for both polarities. This is the point where it is useful not to hide the labels from the question asking device.

If this is a human, he (or she) may perhaps decide — by looking at the labels or better at the input formulae they denote — what was the reason and how the conflict should be resolved. There are possibilities to do this in a manner enabling the system to do the same on its own in future times.

[4]The set of found nogoods even can be increased, if α_1 and α_2 contain common prime implicants.

One could for example retract a formula. This is *not* done by eliminating it from the input data base, as then the whole process has to be started anew and all lemmata generated in the meantime must be eliminated, but can be handled by simply declaring the corresponding label a nogood. This should be done in a separate "withdrawn list" distinguished from the nogood data base, if this process is intended as reversible. So any formula can be switched "on" and "off" simply by adding or deleting its label to/from that list. Besides, there is no reason why the list should only contain atomic nogoods. In fact it can contain any arbitrary label, exactly like the nogood data base. This is a true possibility to describe exclusions of defaults[5].

There is another way of resolving conflicts. It is not necessary to block certain formulae completely, as is done when declaring them nogoods. When there are proofs for $\alpha_1{:}F$ as well as $\alpha_2{:}\neg F$, the weakest condition sufficient for a true decision between yes and no is to declare one of α_1 and α_2 to be bigger according to some ordering than the other. This can be done by refining our partial order for labels. It first is to show that this is a true refinement, i.e. that no inconsistencies occur from doing so. But this is obvious, because if one was bigger than the other in our usual sense, the smaller one is a nogood and this cannot occur, as it would already have been eliminated.

As an aside it should be mentioned, that this construction fits into the paradigm of an *anytime procedure* (Dean & Boddy, 1988; Russell & Zilberstein, 1991)[6].

No let us turn to the proof procedures needed. The problem we have to solve, given a query, differs from the one encountered in automated theorem proving. Finding one proof is not sufficient. Instead we are interesting in all possible proofs, or at least particular representatives. What we need can be reduced to the so-called *consequence finding* problem. This problem is well-known in literature. The name has been coined by Lee (1967). He uses it to clarify the distinction to *proof finding* (Robinson, 1967).

Fortunately, known proof procedures can be used for this problem. Ordinary resolution (Robinson, 1965) for instance is complete for *prime consequences.* A similar result has been obtained for semantic resolution (Slagle, 1967) in (Slagle et al., 1969) and for linear resolution (Anderson & Bledsoe, 1970) in (Minicozzi & Reiter, 1972). Newer results more directly related to our domain of discussion can be found in (Inoue, 1992; Demolombe & Fariñas del Cerro, 1991).

The definition of a prime consequence (also known as non-trivial consequence) is: A *clause* C is a prime consequence of S, if S implies C and there is no consequence D (different from C) of S with C implies D. For propositional logic this is exactly dual to prime implicants, therefore the term prime implicates can also be found[7].

---

[5]In our example we have only given propositional labels. The whole procedure will run the same way with the first order labels we shall encounter in chapter 8, when we talk about defaults.

[6]Similar concepts can be found in (Horvitz, 1989). In our context especially (Haddawy & Frisch, 1991) should be mentioned.

[7]Slagle et al. (1969): "the prime consequence of a set of ground clauses is a *prime implicate* of the set and is the dual of a prime implicant of the dual of the set".

The completeness proof for resolution in consequence finding shows the following: When starting with a *satisfiable* set of axioms, and then doing resolution, we get no complete enumeration of theorems, but for every entailed T there is a derivable T' that subsumes T. This is sufficient for prime consequences.

Application to CMS/ATMS is done in (Inoue, 1990). Inoue explains the importance of consequence finding for the whole area of artificial intelligence (p. 341) and why it has been neglected up to then (p. 302). An improved algorithm (SOL resolution) is given by Inoue (1992)[8].

In practice there is the difference between the *interpreted* versus the *compiled* approach (Reiter & de Kleer, 1987), which means advance computation of prime implicates. Inoue (1990) does this via saturation. From the viewpoint of theory alone there is no difference between those methods. The enumeration problem of prime implicates is of exponential complexity (Inoue, 1990).

An interesting resolution strategy for ATMS problems (directed CAT-correct resolution with clash, RCD) can be found in (Tayrac, 1990) resp. (Cayrol & Tayrac, 1989).

Our abstract reasoner bears a lot of similarities with the "belief subsystem" in (Konolige, 1983). There also the resource boundedness of computation is taken into account, so that Konolige comes up with effectively computable, sound inference rules in a system consisting of the inference rules, a base set, a control strategy (!), queries and answers. The answers are, however, simply "yes" or "no". Konolige writes:

> "In particular, this forces deduction rules to be *monotonic*. It is our view that nonmonotonic or default reasoning should occur in the belief updating and revision process, rather than in querying beliefs."

Similarly to our approach, the filtering goes extra and can in principle be done in the end. However, to accelerate computation, a lot can be computed in advance, as in the case of nogoods. This works like the restriction sets of Martins and Shapiro.

The main difference between our method and Konolige's is his totality demand:

> "the answer to a query will be returned in a finite amount of time"

With "answer" Konolige means the final yes/no decision. This demand is weakened in our approach to that in a finite amount of time *an* answer is given. This answer will at any time contain a label correct wrt. LL-entailment, but that label need not be consistent. With increasing time, the label will suffice ω-correctness for also increasing ω and thus approach consistent LL-entailment. If only the yes/no answer is considered, this may be wrong, and in particular

[8] In that same article a generalization of characteristic clauses is given, so that CMS/ATMS as well as abduction turn out to be special cases of the same problem. There are similar results in (Demolombe & Fariñas del Cerro, 1991).

may toggle between yes and no several times. This seems horrible at first view, but it comes rather close to how humans deal with that matter (given more time more justifications as well as more rejections for a conjecture will come to one's mind). Besides one could not seriously expect to ever yield better results for undecidable logics.

Konolige talks about the difference between *deductive consistency* and *logical consistency*. Deductive consistency means there may be logical consistency, but the system *will not be able to detect it*, thus the matter is consistent wrt. to the system given. Instead of restraining ourselves to deductive consistency, as Konolige proposes, we stick to logical consistency, as our systems are in principle able to detect any inconsistency, given infinite amounts of time, but allow tentative answers, which however can be characterized correct wrt. to the approximative semantics presented.

There are also other system proposals in the literature on data (or knowledge) base updates that resemble the abstract reasoner in certain points. We only mention (Winslett, 1986; Fagin, Ullman & Vardi, 1983; Fagin, Kuper, Ullman & Vardi, 1984; Abiteboul & Grahne, 1985; Imielinski & Lipski, 1984; Levesque, 1984*a*; Reiter, 1984; Winslett-Wilkins, 1986) without discussing them. The interface construction goes back to (Levesque, 1984*b*), who termed it TELL/ASK. In Levesque's original article the TELL operation is however very restricted. As he only eliminates models, anything introduced has to be consistent with the old state.

Chapter 8

Default Handling

Within this chapter we sketch how our approach can be used to describe default reasoning as well. For this task we have to extend the logic of the labels to first order logic. In contrast to the sections on Reason Maintenance we do not claim to possess a framework capable of describing all the approaches proposed up to now. In fact we simply introduce our way of handling defaults and only afterwards compare it to the existing ones. It will turn out that we can gain some interesting insights on default logics.

8.1 Motivation: The Flying Birds Example Revisited

Seemingly all kinds of default logics are motivated by the need to model *typicality*. One of the standard examples goes like this

Example 8.1.1 (Does Tweety fly?[1])

Typically birds fly
Penguins are birds
Penguins don't fly

If we now are told, that Tweety is a bird, then we should come to the conclusion that Tweety flies, as any bird should be assumed a typical bird, as far as there are no hints to the contrary. If, however, we get to know instead that Tweety is a penguin, we are no more allowed to assume him to fly, as — though he certainly remains a bird – the fact that penguins do not fly is *hard*, as compared to the *default* that birds typically fly.

There are many approaches that tackle this kind of reasoning. Most are based on a dichotomy within the rules of the calculus used. There are some rules that can always be applied, whereas others, the so-called *default rules* are only

[1]This example is in fact very old. In a slightly different form it can be found already in Frege (1879, page 51).

applicable if certain conditions are met. These conditions are almost always not testable in a *constructive* way. By this we mean that the test for applicability is often based on knowledge that is available only after the application itself, i.e. it is possible that the result can tell that the rule should not have been applied at all.

In Reiter's logic DL (Reiter, 1980*b*) for instance default rules are written as[2]

$$\frac{\alpha : M\beta}{\gamma}$$

with the intended meaning "if α is derivable and it is consistent to assume β, then γ is derivable per default".

More formally:

Definition 8.1.2 (Reiter Default)
A (Reiter) **default** is of the form

$$\frac{\alpha[\overline{x}] : M\beta[\overline{x}]}{\gamma[\overline{x}]},$$

where α, β and γ are first order formulae with common free variables $\overline{x} = \mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}$. α is called the **prerequisite**, γ the **consequent**.

A rather uninteresting, although well examined, subclass of defaults are closed defaults:

Definition 8.1.3 (Closed Default)
A default is called **closed**, if it does not contain free variables. A default that is not closed is called **open**.

Definition 8.1.4 (Default Theory)
A **default theory** consists of a set F of closed first order formulae and a set Δ of defaults. A default theory is called **closed**, if all elements of Δ are closed.

The semantics for DL talks about *extensions*.

Definition 8.1.5 (Extension)
Let (F, Δ) be a default theory. Then an **extension** is a fixed point of the Γ operator, which maps sets of *closed* first order formulae (wffs) on sets of wffs and is defined as: $\Gamma(S)$ is the smallest set with

- $F \subseteq \Gamma(S)$
- $\Gamma(S)$ is closed wrt. (classical) logical consequence,
- for every default rule $\alpha : M\beta/\gamma \in \Delta$, if $\alpha \in \Gamma(S)$ and $\beta \notin S$, then $\gamma \in \Gamma(S)$.

Reiter gives an algorithm that computes extensions by the iteration

[2]within text we write $\alpha : M\beta/\gamma$ instead.

Algorithm 8.1.6

$$
\begin{aligned}
E_0 &= F \\
E_{i+1} &= \mathsf{Cons}(E_i) \cup \{\gamma \mid \alpha : M\beta/\gamma \in \Delta, \alpha \in E_i, \beta \notin E\}.
\end{aligned}
$$

The extension is then defined as

$$E = \bigcup_{i=0}^{\infty} E_i.$$

This algorithm is, however, not constructive, for E has to be guessed beforehand (note the occurrence of E in the second equation). So checking whether a given set E is in fact an extension can be done with the help of this algorithm, but extensions can not be constructed.

We shall approach that matter in a slightly different way. Instead of forbidding to use a rule because it will contribute to a contradiction not known at the moment, we *admit* that step and only later, when we really find out about a contradiction, eliminate some consequences. The advantage is that thus we can proceed incrementally and use inference rules that are only locally informed. Instead we have the difficulty to find an appropriate notion of soundness for the intermediate results.

This looks very much the same as our incremental calculi presented in the preceding chapter. And in fact it bears the same spirit and only one rather simple generalization has to be performed to obtain the possibility of modelling default reasoning.

Let us try to model the example of the non-flying penguins. We can see the statements "all penguins fly" and "penguins are birds" as axioms, as well as perhaps "Tweety is a penguin". Like in the modelling of an ATMS we label them with $\top$.

In contrast, "typically birds fly" is something like an assumption. Assumptions and defaults have in common, that both may be used only as long as their use does not produce trouble. In a first approach we could try to label the formula $\forall x \mathrm{BIRD}(x) \rightarrow \mathrm{FLIES}(x)$ with an atomic label, thus modelling the complete example as

$$
\begin{gathered}
\alpha{:}\forall x \mathrm{BIRD}(x) \rightarrow \mathrm{FLIES}(x) \\
\top{:}\forall x \mathrm{PENGUIN}(x) \rightarrow \mathrm{BIRD}(x) \\
\top{:}\forall x \mathrm{PENGUIN}(x) \rightarrow \neg\mathrm{FLIES}(x)
\end{gathered}
$$

adding $\top$:BIRD(Tweety) or $\top$:PENGUIN(Tweety). If we now look for consistent consequences, we find that in the first case α:FLIES(Tweety) is maximally consistently derivable, while in the second case α is a nogood and $\top$:$\neg$FLIES(Tweety) follows consistently, which looks rather like what we intended.

There is just one flaw with this approach. If in addition to

$$\top\text{:PENGUIN(Tweety)}$$

we added

$$\top\text{:BIRD(Hansi)},$$

we can not derive

$$\alpha\text{:FLIES(Hansi)},$$

as α is a nogood.

What we have done is simply reject the whole formula $\forall x \text{BIRD}(x) \rightarrow \text{FLIES}(x)$ because of the counterexample. Instead in most default logics the defaults are not treated as formulae, but rather as *schemata*, that stand as representatives for all their ground instances.

To get a similar effect, we enhance the logic of the labels. We permit them to be first order formulae, intending the following representation of the problem:

$$\forall x\ \alpha(x)\text{:BIRD}(x) \rightarrow \text{FLIES}(x)$$
$$\top\text{:}\forall x \text{PENGUIN}(x) \rightarrow \text{BIRD}(x)$$
$$\top\text{:}\forall x \text{PENGUIN}(x) \rightarrow \neg\text{FLIES}(x)$$

If we design a calculus the way that instantiation of those variables in the formula part that also appear in the labels, is carried over to the labels, we can diagnose e.g. $\alpha(\text{Tweety})$ as a nogood by addition of $\top$:PENGUIN(Tweety), thus preserving the possibility to derive the formula $\alpha(\text{Hansi})$:FLIES(Hansi) from $\top$:BIRD(Hansi) without violating consistency.

What we shall do in the rest of the chapter is the following. First we give a very brief introduction to some of the main approaches for dealing with defaults. This mainly serves the purpose of supplying the terminology needed. Then we work out our representation introduced rather informally up to now in a more accurate way. Afterwards we compare our approach with some other known default handling methods. We close the chapter by dedicating a whole section on a particular famous problem, which we shall be able to shed some additional light on with the help of our formalism.

8.2 Default Logics: Approaches and Terminology

In (McDermott & Doyle, 1980) the relation between formal logics ("laws of thought", the "ideal mind") and the *operation* of the mind is discussed. This leads to a modal approach using the operator M, which is meant to denote "is consistent", thus introducing *defeasible* reasoning. The resulting logics is not semi-decidable, but *asymptotically decidable*, which means the procedure changes its answer only a finite number of times[3].

[3]The same holds for our incremental calculi.

> "a procedure of this kind could be useful in spite of the provisional nature of its outputs, since a robot always has to act on the basis of incomplete cogitation"

Their logic is known under the name NML I.

The M operator appears again in Reiter's default logic (Reiter, 1980*b*), which we already came across in the introductory section. As DL is probably the best known default mechanism, we will use terminology borrowed from there.

In Reiter's logic, a formula follows (from facts F and default rules Δ) by default, if it is contained in at least one extension. But unfortunately not every default theory does have an extension. This is quite inconvenient. So one considers restricted cases of default theories that guarantee existence of an extension. In this context the following definitions arose.

Definition 8.2.1 (Types of Default Rules)
A default rule

$$\frac{\alpha : M\beta}{\gamma}$$

is called a **normal** default, if $\beta = \gamma$. It is called a **semi-normal** default, if $\beta \vdash \gamma$. We call a default with empty (missing) α **prerequisite-free**.

Reiter was able to supply a proof theory for closed normal default theories, which he extended to open normal default theories[4]. Normal default theories are guaranteed to always possess an extension. The problem, whether some formula is in an extension, is known to be not even semi-decidable.

Some other approaches to default reasoning use modal logics, such as (McDermott, 1982*a*; McDermott, 1980; Moore, 1983; Moore, 1985; Marek & Truszczyński, 1990). (Moore, 1985) proposes the terminology *autoepistemic logic* (AEL) instead of default logic, as he sees the main characteristics of the problem given by the introspection aspect. The same should hold for (McDermott, 1982*a*) and (McDermott & Doyle, 1980).

Still another direction are different kinds of *circumscription*, like (McCarthy, 1980; McCarthy, 1984; Lifschitz, 1985; Lifschitz, 1986), an idea that is actually older than defaults, as predicate completion already appears in (Clark, 1978).

There is an article (Sombé, 1990) that compares the various approaches using one single example for all of them.

The aproach that most resembles ours is Poole's. Poole (1985) (and (Poole, 1988*a*))[5] uses abduction to describe the matter. Abduction is a form of hypothetical reasoning which tries to find *explanations* for found observations, that means the direction of inferences is from conclusion to premises. An introduction can be found in (Levesque, 1989)[6]. Logical characterizations of abduction

[4] Open defaults have recently been detected to cause problems when doing Skolemization (Baader & Hollunder, 1992). The procedure of Junker & Konolige (1990*b*) (for decidable base logic) does only work for closed defaults. But the interesting ones are indeed the open ones, as already mentioned by Reiter (1980*b*).

[5] The THEORIST system is described in (Poole, 1984)

[6] It is interesting that Levesque explicitly mentions the tight connection to ATMS.

are given in (Eshghi & Kowalski, 1988; Poole, 1988*b*; Reiter, 1987). The last two mentioned introduce the list of marked possible hypotheses, called the **abducibles**.

Poole's defaults are those predefined hypotheses (abducibles), and an additional consistency check is done. There is a distinction between hard **facts** (F) and a pool of possible hypotheses (**defaults**, Δ), a proposition g being entailed (in Poole's terminology: is *explainable*), if there exists a set D of ground instances of elements of Δ, such that

$$F \cup D \models g$$

and

$$F \cup D \quad \text{is consistent.}$$

Definition 8.2.2 (Scenario)
A **scenario** of F, Δ is a set $D \cup F$, where D is a set of ground instances of elements of Δ such that $D \cup F$ is consistent.

Definition 8.2.3 (Explanation)
If G is a closed formula, then an **explanation** of G from F, Δ is a scenario of F, Δ that implies G.

Poole admits as defaults (elements of Δ) arbitrary first order formulae. In Reiter's terminology this means admittance of open defaults.

Poole also gives a definition for extensions that is different from Reiter's, but are related to them.

Definition 8.2.4 (Extension)
An **extension** of F, Δ is the set of logical consequences of a maximal (wrt. set inclusion) scenario of F, Δ.

Theorem 8.2.5 (Poole (1988*a*))
Let E be an extension. Then

- $F \subseteq E$
- $\mathsf{Cons}(E) = E$
- *if γ is a ground instance of an element of Δ and $\neg\gamma \notin E$, then $\gamma \in E$.*

Furthermore E is minimal with respect to the above three properties.

8.3 First Order Labels for Defaults

In this section we show how the definitions and theorems of chapter 4 can be generalized so as to handle first order labels. So from now on we build our labelled formulae in the following way: The right part is taken from classical first order logic as basic logic, whereas labels are now also classical first order predicate logic formulae. We can assume all our formulae to be closed by implicit universal quantification. So e.g. $\alpha(\overline{x}){:}F(\overline{x})$ is actually $\forall\overline{x}\ \alpha(\overline{x}) \rightarrow F(\overline{x})$, where $\overline{x}$ represents a vector of variables. However, we usually do not write the quantifier but denote the formulae with free variables, in order to emphasize that if we speak of **instances** of labelled formulae, we only want to instantiate these free variables. So e.g. $\alpha(a){:}\forall y\ F(a,y)$ is an instance of $\alpha(x){:}\forall y\ F(x,y)$, whereas $\alpha(a){:}F(a,b)$ is not. Intuitively, like in most default languages, such open formulae stand as representatives for all of their instances.

Definition 8.3.1 (Labelled Formulae and Substitutions)
A **labelled formula** is of the form $L[\overline{x}]{:}F[\overline{x}]$, where F is a first order formula with free variables $\overline{x}$ and L a first order formula with $\overline{x}$ as its only variables.

If $G = \alpha{:}F$ is a labelled formula, then the **substitution** σG is defined as $\sigma G = \sigma\alpha{:}\sigma F$.

Most definitions of chapter 4 carry over, so we give only the important ones briefly. We have to be more precise about what we mean by DDNF in this context: According to the fact that non-ground atoms stand for their instances and therefore in particular for all their ground instances, they denote in fact *sets of ground instances* and can be viewed as shorthand for writing these down (of course this is not possible, for there are usually (countably) infinitely many of them). So the DDNF formation can be done on the ground atoms and the result again notated in shorthand. As an example, $\alpha(f(x))\alpha(x)$ can be shortened to $\alpha(x)$.

Definition 8.3.2 (Basic and Semi-basic Sets)
A set of labelled formulae is called **basic**, iff every label is atomic and has an arity corresponding to the number of variables free in the formula part, which occupy the argument positions in the label. Furthermore no label occurs more than once. It is called **semi-basic**, if in addition the occurrence of arbitrarily many formulae labelled $\top$ is allowed. The formulae labelled $\top$ are called **facts**, the others **defaults**.

Definition 8.3.3 (Positive and Relevant Labels)
A label is called **positive** if its DDNF does not contain negative literals. It is called **relevant** wrt. a given semi-basic set Φ of labelled formulae, if each of its literals is an instance of a label of some formula in Φ. A labelled formula is said to be relevant resp. positive if its label is.

Definition 8.3.4 (Characteristic Function)
Let Φ be a semi-basic set of labelled formulae, and let $\alpha(\mathrm{t}_1, \ldots, \mathrm{t}_\mathrm{n})$ be an atomic relevant label. Then there exists a unique formula $\alpha(\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}){:}F$ in Φ such that $\alpha(\mathrm{t}_1, \ldots, \mathrm{t}_\mathrm{n})$ is an instance of $\alpha(\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n})$. Now let $\mu(\alpha)$ denote the set

$$\{\sigma F \mid \sigma(\alpha(\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}){:}F) \text{ is an instance of } \alpha(\mathrm{x}_1, \ldots, \mathrm{x}_\mathrm{n}){:}F\} \cup \{G \mid \top{:}G \in \Phi\}.$$

Then for positive, relevant α the characteristic function χ is defined as

$$\chi(\alpha) = \begin{cases} \{\} & \text{if} \quad \alpha = \bot \\ \{\{\}\} & \text{if} \quad \alpha = \top \\ \bigcup_{i=1}^{m} \{ \bigcup_{j=1}^{n} \mu(a_{ij})\} & \text{if} \quad \alpha = \sum_{i=1}^{m} \prod_{j=1}^{n} a_{ij} \end{cases}$$

(where the $\sum$ and $\prod$ stand for disjunction resp. conjunction) if α is in DDNF, else $\chi(\alpha) = \chi(\mathrm{DDNF}(\alpha))$.

Remark 8.3.5
The characteristic function always yields a set of instances of $\mathsf{formula}(\Phi)$.

Definition 8.3.6
If $G = \alpha{:}F$ is a labelled formula, then $\mathsf{label}(G) = \alpha$. Let Ψ be a set of instances of elements of a set Φ of labelled formulae. Then define

$$\mathsf{label}(\Psi) = \bigwedge_{\varphi \in \Psi} \mathsf{label}(\varphi)$$

Definition 8.3.7 (Logical Consequence on Labelled Formulae)
Let Φ be a set of labelled formulae and $\alpha{:}F$ a single labelled formula. We say $\alpha{:}F$ **follows** (logically) from Φ (written as $\Phi \models_{\mathrm{LL}} \alpha{:}F$), iff there exist subsets $\Psi_1, \ldots, \Psi_{\mathrm{n}}$ $(n \geq 0)$ of the set of all instances of Φ, with

- $\forall\, \Psi_i :\ \mathsf{formula}(\Psi_i)\ \models_{\mathrm{BL}} F$ and
- $\models_{\mathrm{FOL}} \alpha \rightarrow \bigvee_{i=1}^{n} \mathsf{label}(\Psi_i)$.

The ordering relation on labels is generalized in the obvious way, in that instances are *greater* than their generalizations.

Example 8.3.8
The set of labelled formulae

$$\alpha(x){:}\mathrm{BIRD}(x) \rightarrow \mathrm{FLIES}(x)$$

$$\top{:}\mathrm{BIRD}(tweety)$$

entails e.g. $\alpha(tweety){:}\mathrm{FLIES}(tweety)$ as well as $\alpha(x){:}\mathrm{FLIES}(tweety)$. The first of these labels is the maximal label for $\mathrm{FLIES}(tweety)$.

Definition 8.3.9 (Consistent LL-Entailment)
Let Φ be a set of labelled formulae and $\alpha{:}F$ a single labelled formula. We say $\alpha{:}F$ **consistently follows** from Φ (written as $\Phi \models_{\mathrm{consLL}} \alpha{:}F$), iff

- $\Phi \models_{\mathrm{LL}} \alpha{:}F$ and
- for all prime implicants β of α holds $\Phi \not\models_{\mathrm{LL}} \beta{:}\bot$.

Now we can give our translation from (Poole) defaults to labelled logic. As can be expected, facts are mapped to facts in our sense and defaults to defaults.

Definition 8.3.10 (Labelled Default Theories)
Let $D = (F, \Delta)$ be a default theory. Then its **labelled representation** is a set D_{LL} of labelled formulae obtained by translating

- every fact $f \in F$ to $\top{:}F$,

- and every default $d[\overline{x}] \in \Delta$ to $\alpha(\overline{x}){:}d[\overline{x}]$, where α is a new predicate.

We then get as a result

Theorem 8.3.11
Given a default theory D according to Poole and its labelled representation D_{LL}. If a formula G is explainable in D, then there exists a label α such that $D_{\mathrm{LL}} \models_{\mathrm{consLL}} \alpha{:}G$.

Proof:
This is immediately obvious. A formula G is Poole explainable exactly if there exists a set Δ' of ground instances of elements of Δ, such that $\Delta' \cup F$ entails G and $\Delta' \cup F$ is consistent. This is exactly modelled by our translation. □

The other direction is slightly more complicated, for we can deduce *more* in the translation. This comes from the fact that Poole (like others) allows only *ground* instances of defaults, whereas we admit arbitrary instances.

Example 8.3.12
From

$$\alpha(x){:}\mathrm{BIRD}(x) \rightarrow \mathrm{FLIES}(x)$$

$$\top{:}\mathrm{PENGUIN}(x) \rightarrow \mathrm{BIRD}(x)$$

we can deduce

$$\alpha(x){:}\mathrm{PENGUIN}(x) \rightarrow \mathrm{FLIES}(x),$$

which is not possible in Poole's approach. However, for arbitrary ground terms t

$$\mathrm{PENGUIN}(t) \rightarrow \mathrm{FLIES}(t)$$

is explainable.

So we have

Theorem 8.3.13
Given a default theory D according to Poole, its labelled representation D_{LL} and an unlabelled formula G. If there exists a label α such that $D_{\mathrm{LL}} \models_{\mathrm{consLL}} \alpha{:}G$, then every ground instance of G is explainable in D.

Proof:
From the definition of χ and the label function. □

So we have shown that our approach is a generalization of Poole's. We want to emphasize that the incremental calculi from chapter 7 can also be generalized so as to work with first order labels. We only have to make sure that every instantiation in the formula part occurring in the derivation process must also be applied to the labels, so that rules of the form

$$\frac{F_1, \ldots, F_n}{\sigma G}$$

are now mapped to

$$\frac{\alpha_1{:}F_1, \ldots, \alpha_n{:}F_n}{\sigma(\alpha_1, \ldots, \alpha_{\mathrm{n}}{:}G)}.$$

8.4 How is Our Approach Related to Other Default Logics?

As already seen, there are various different approaches to handling defaults. They often yield different results for the same problem. So there has been much discusion of which approach can be viewed as adequate, and in particular, whether it is really necessary to leave first order logic for this.

Poole (1988*a*), for example, strongly objects the necessity of a special logic beyond ordinary first order logics for modelling default reasoning:

> "there is nothing wrong with classical logic; we should not expect reasoning to be just deduction from our knowledge. (...) defaults are *implicit assumptions*; we have to make these assumptions explicit (...)"

> "I argue that, rather than being a problem with logic, nonmonotonicity is a problem of how logic is used"[7]

The simplest case is described as

> "the user provides the form of possible hypotheses they are prepared to accept in an explanation"

In his case this "form" seems to mean open defaults.

> "Rather than defining a new logic for default reasoning, we would rather say that it's the natural outcome of considering reasoning, not as deduction, but as theory formation. It is logic which tells us what our theory predicts."

We go one step further in that we look upon the whole thing as deduction as well, thus incorporating some metalevel reasoning into the object level, true to the spirit of LDS.

> "if one allows hypothetical reasoning, then there is no need to define a new logic to handle nonmonotonic reasoning"

Here the author wholeheartedly agrees. As can be seen in the section on fibering semantics (4.9) it is possible to view our labelled formulae, though they are presented differently, as ordinary first order formulae.

Like our approach, Poole's is — though presented as first order — in fact *independent of the particular logic used*. Crucial is his definition of a *scenario*: D is a ground instance of a subset of Δ, which is consistent. An extension is then a maximal (wrt. set inclusion) scenario. What is inherited of first order semantics is e.g. compactness. If something is explainable, then it is explainable with a finite scenario. In our case that corresponds to the fact that maximal labels are finitely representable.

[7] To this theme cf. (Israel, 1980).

Comparison to DL shows, that Poole's defaults are normal.

Poole names his defaults. The names themselves are parametrized by the free variables in the default. This is exactly what we get in our case. Poole explains the intuitive meaning as

> "have the name implying the default as a fact"

There is an interesting theorem in (Poole, 1988*a*) (theorem 5.1, p. 33) that states that any default theory can be transformed into an equivalent one, where Δ contains only the names of defaults, whereas for every default $d \in \Delta$ a formula name $\to d$ is added to the set of facts. This is even closer to our representation than the original version.

Like Poole, our approach admits open defaults, but is restricted to normal ones. Poole gives a possibility to translate arbitrary Reiter defaults:

$$\frac{\alpha(x) : M\beta_1(x) \ldots M\beta_n(x)}{\gamma(x)}$$

becomes

$$\gamma(x) \leftarrow M\beta_1(x) \wedge \ldots \wedge M\beta_n(x) \wedge \alpha(x)$$

with an additional "constraint"

$$\neg M\beta_i(x) \leftarrow \beta_i(x).$$

Poole's definition of constraints is to introduce a set C of first order formulae, that are used when checking for consistency, but not for the explanation. This means a formula G is explained by a default theory (F, Δ, C), if there is a set Δ' of ground instances of elements of Δ, such that $\Delta' \cup F$ entails G and $\Delta' \cup F \cup C$ is consistent.

This can be modelled in our aproach by redefining consistent LL-entailment as to also use an additional set of formulae for the derivation of $\perp$, which does not present any problem.

Poole also presents a way to model how one can explicitly state that one default blocks another. This is done by adding to the facts formulae containing the names of the involved defaults. This can not be directly modelled by our approach, because our labelled formulae are all of a particular form, namely the label implies the formula part. Digression from this is in principle possible, as long as only semantics is concerned, because we always can interpret the complete labelled formula as a simple first order formula with ":" as $\to$. But of course we can then not deal with the labels separately, as in the proposed calculi.

Coming back to the question of what method of treating defaults is adequate, the author thinks that there is a strong argument for approaches like Poole's, and therefore ours, too. There is an interesting article by Brewka (1989) that describes the nature of a default (as compared to a fact).

> "What makes a default a default? What distinguishes it from a fact? Certainly our attitude towards it in case of conflict, i.e. an inconsistency. If we take this view serious then the idea of default reasoning as a special case of inconsistency handling seems quite natural."

He stresses that the difference becomes apparent only if contradictions arise[8]. The idea of a *maximal consistent subtheory* goes back as far as to Rescher (1964).

8.5 Some Thoughts on the Anomalous Extension Problem

In this section we shall discuss a famous problem well known from literature. Opinions whether the correct solution to it has already been found, differ. The author wants to put forth his own opinion on this issue, arguing that our framework rather clearly shows how the solution should look like.

Being a little bit naïve, one expects that default reasoning should solve things like the *frame problem* (McCarthy & Hayes, 1969) or the phenomena of *persistence* (McDermott, 1982*b*), as these have been the very reasons for its introduction.

Unfortunately this is not true. As early as in (Reiter & Criscuolo, 1981) it was noticed that there are problems resulting from the interaction of defaults and that certain extensions should be excluded. In contrast to the conjecture in (Reiter, 1980*b*) this involves using *non normal defaults.* The attempt to restrict the class of defaults needed, *semi-normal* theories were introduced, but the properties these exhibit are not particularly appealing (no existence guarantee for extensions, no semimonotonicity, no proof theory)[9].

A lot of people's hopes were then destroyed by the famous article of Hanks & McDermott (1986) (this is an improved version of their original article (Hanks & McDermott, 1985)[10]). The sad message simply is, that DL does not do what is intuitively expected.

> "On one hand the logics have been subjected to intense technical scrutiny (...) and have shown to produce counterintuitive results under certain circumstances. At the same time we see in the literature practical representation problems (...) in which default rules would *seem* to be of use, but in these cases technical details of the formal systems are for the most part ignored.

[8] A similar remark is due to Bibel (1985).

[9] There is an interesting note on integrity constraints in (Reiter, 1980*b*) (p. 275). These can not be incorporated that easily into DL. Our approach pursues that direction again: First contradiction is permitted, and only afterwards the user is *asked* how inconsistencies are to be resolved (by supplying a partial order on labels).

[10] There is also a later version (Hanks & McDermott, 1987) which contains the authors' answers to several objections.

> The middle ground — whether the technical workings of the logics correctly bear out one's intentions in representing practical default-reasoning problems — is for the most part empty ...".

What follows then is known as the *Yale shooting problem*, which is not properly solved by DL as well as by circumscription (McCarthy, 1980).

The situation is described using *situation calculus*. There are predicates that hold (or do not hold) in certain *situations*. Furthermore there are *actions* that transform one situation into another. The effect of actions is described as enumerations of the predicates that hold in the result situation. The *frame problem* is now that this enumeration is lengthy if it should be exhaustive, even though most actions leave most predicates absolutely unchanged, which is called *persistence*.

With defaults this can be modelled by stating the persistence axioms as defaults.

The shooting example consists of a situation s_0 and three following actions.

- First, in s_0 a gun is loaded. That yields situation s_1.
- Then the gunner waits some time. Waiting is another action and here leads to situation s_2.
- In s_2, finally, the gunner shoots at his victim.

Giving the facts that loading causes a gun to be loaded and that shooting a loaded gun on somebody causes his death (of cause this is simplified), the victim should be dead in s_3. But we do not know if the gun is loaded in s_2. We can only assume this because of persistence and the implicit assumption that waiting does not change anything. With the same reason we could, however, assume that the victim is still alive in s_3 because of the persistence of staying alive. Of course this has to be blocked. This can be done by explicitly telling that shooting a loaded gun at someone is exceptional for this person's staying alive. A formulation of the complete story (reduced to the important things) in a first order representation, where the predicates $t(p,s)$ and $ab(p,a,s)$ mean "predicate p is is true in situation s" resp. "in situation s the action a is an exception to the persistence of predicate p" and the function $\text{res}(a,s)$ denotes the resulting situation from performing action a in situation s, is then:

Example 8.5.1 (Yale Shooting Example, Circumscriptive Version)

$$t(\text{alive}, s_0)$$

$$\forall s\ t(\text{loaded}, \text{res}(\text{load}, s))$$

$$\forall s\ t(\text{loaded}, s) \to ab(\text{alive}, \text{shoot}, s) \wedge t(\text{dead}, \text{res}(\text{shoot}, s))$$

$$\forall s\ t(\text{alive}, s) \leftrightarrow \neg t(\text{dead}, s)$$

$$\forall p, a, s\ t(p, s) \wedge \neg ab(p, a, s) \to t(p, \text{res}(a, s))$$

By circumscribing the *ab* predicate one can now hope to obtain the result wished for. But, as it turns out, there do exist *two* minimal models. One is the one we hoped for, but there is another one in which the victim is still alive in s_3, whereas the gun mysteriously ceased to be loaded during the waiting.

This is no fault of circumscription[11]. DL translations of the same problem, as well as various others, all yield the same result. Using DL-terminology one speaks of an *anomalous extension*.

The conclusion of Hanks and McDermott is

> "...we need to re-evaluate the relationship between nonmonotonic logics and human default reasoning. We can no longer engage in the logical 'wishful thinking' that led us to claim that circumscription solves the frame problem ..."

As a reaction several solutions have been proposed in turn. Examples are

- the problem is caused by an incorrect modelling of time (examples are (Shoham, 1986; Shoham, 1988)).
- other formalisms such as pointwise circumscription (Lifschitz, 1986)

According to the author's opinion the first argument is simply not true, as the problem also occurs in examples not involving temporal reasoning, which has been demonstrated e.g. in (Morris, 1988*a*).

For the discussion let us look at a formulation of the problem within our framework. This suffers from the same problem, i.e. here, too, we get the second extension. But we argue, that this *should* be so, for there is nothing false with the unwanted extension. We argue that our disliking that extension is based on an assumption not expressed in the formulation of the problem, so we simply can not expect any reasonable calculus to come up with the wanted result only.

Our formulation is very much like the circumscription formulation, the only difference being that instead of circumscribing the abnormality predicate we introduce the explicit assumption that there are no abnormal situations.

Example 8.5.2 (Yale Shooting Example, LL-Version)

$$\top{:}t(\text{alive}, s_0)$$

$$\top{:}t(\text{loaded}, \text{res}(\text{load}, s))$$

$$\top{:}t(\text{loaded}, s) \rightarrow ab(\text{alive}, \text{shoot}, s) \wedge t(\text{dead}, \text{res}(\text{shoot}, s))$$

$$\top{:}t(\text{alive}, s) \leftrightarrow \neg t(\text{dead}, s)$$

$$\top{:}t(p, s) \wedge \neg ab(p, a, s) \rightarrow t(p, \text{res}(a, s))$$

$$\alpha(p, a, s){:}\neg ab(p, a, s)$$

[11] Nor is the situation calculus to blame for that. There are other formulations of the problems that show exactly the same effect.

Using the shorthand notations

$$\begin{aligned} s_1 &= \mathrm{res}(\mathrm{load}, s_0) \\ s_2 &= \mathrm{res}(\mathrm{wait}, s_1) \\ s_3 &= \mathrm{res}(\mathrm{shoot}, s_2) \end{aligned}$$

we can deduce

$$\alpha(\mathrm{alive}, \mathrm{load}, s_0){:}t(\mathrm{alive}, s_1)$$

$$\top{:}t(\mathrm{loaded}, s_1)$$

concerning situation s_1.

Proceeding further to s_2 we get

$$\alpha(\mathrm{alive}, \mathrm{load}, s_0)\alpha(\mathrm{alive}, \mathrm{wait}, s_1){:}t(\mathrm{alive}, s_2)$$

$$\alpha(\mathrm{loaded}, \mathrm{wait}, s_1){:}t(\mathrm{loaded}, s_2)$$

Next comes s_3 and now it becomes interesting:

$$\alpha(\mathrm{alive}, \mathrm{load}, s_0)\alpha(\mathrm{alive}, \mathrm{wait}, s_1)\alpha(\mathrm{alive}, \mathrm{shoot}, s_2){:}t(\mathrm{alive}, s_3)$$

$$\alpha(\mathrm{loaded}, \mathrm{wait}, s_1){:}t(\mathrm{dead}, s_3)$$

From this we see that

$$\alpha(\mathrm{alive}, \mathrm{load}, s_0)\alpha(\mathrm{alive}, \mathrm{wait}, s_1)\alpha(\mathrm{alive}, \mathrm{shoot}, s_2), \alpha(\mathrm{loaded}, \mathrm{wait}, s_1)$$

is a nogood. But this does not help us very much, since there are obviously consistent labels for $t(\mathrm{alive}, s_3)$ as well as for $t(\mathrm{dead}, s_3)$.

We have not used the axiom that states that shooting a loaded gun at somebody is an abnormality concerning the persistence of living on. Will that solve our problems?

We can deduce

$$\alpha(\mathrm{loaded}, \mathrm{wait}, s_1){:}ab(\mathrm{alive}, \mathit{shoot}, s_2)$$

which shows that we can give a better (tighter) nogood, namely

$$\alpha(\mathrm{alive}, \mathrm{shoot}, s_2), \alpha(\mathrm{loaded}, \mathrm{wait}, s_1)$$

So in our representation we can directly see what is happening: Assuming one normality (either the persistence of staying alive after the shoot or the persistence of the gun staying loaded during the waiting) contradicts the other. This exactly corresponds to what is expressed in all the formulations. So why should one extension be preferable to the other?

One could argue (and it has indeed been) that the example explicitly states that staying alive behaves abnormally in the shooting situation, whereas nothing is told that endangers the assumption of the persistence of staying loaded.

But this is not true. The abnormality wrt. staying alive after shooting has as a premise the statement that the gun is loaded. So assuming the persistence of alive it can be concluded that the gun is not loaded at s_2 and therefore we get the abnormality in the persistence of loaded. There is nothing mysterious here. The author's interpretation of the example is simply that most people implicitly assume that staying alive is at least questioned if shooting occurs, *independently of whether the gun is loaded or not.*

So a better formulation should be axiomatized as

$$\top{:}t(\text{loaded}, s) \rightarrow t(\text{dead}, \text{res}(\text{shoot}, s))$$

$$\top{:}ab(\text{alive}, \text{shoot}, s)$$

which indeed leads to only one extension.

What we want to say is that there is nothing wrong with calculi that yield the second extensions, because its exclusion is not contained in the information the problem formulation, but comes in via particular assumptions of the observer not explicitly stated. Of course there exist different opinions on that topic. In fact nearly every proposal goes into the direction of excluding the unwanted extension. We do not discuss special formalisms, as e.g. pointwise circumscription (Lifschitz, 1986; Lifschitz, 1987), here. Instead we want to discuss only one proposal for a solution that fits well in what we talked about earlier.

Some authors propose that the *direction* of argumentation should be restricted. E.g. Kautz (1986) thinks the reasoning should primarily work *forward in time.* He uses preferred models in a circumscriptive approach. Models are preferred if they delay the assumption of exceptional cases (no persistence) *as late as possible.* This solves the given example, but Kautz himself gives another example that leads this argumentation ad absurdum: Given I parked my car in the morning and now notice it is no more there, I certainly have no reason to prefer the hypothesis it has been stolen just the moment before. The theft may as well have happened immediately after I left my car. He concludes that his approach obviously does not work satisfactorily in all examples, but very often is appropriate, for the true reason seems to be not time, but *causation*, which physically works forward in time.

That time is not the problem, can be seen from the fact that the same phenomena in contexts that have no connection to time (Morris, 1988*a*). In some sense, however, the Morris proposal goes into the same direction. In (Morris, 1987) he reports the Yale shooting problem as solved. He wants to restrict the direction in which the implications are to be read. If this goes "forward" only (this time "forward" refers to the direction of the implication), the example is solved. He therefore proposes to use a TMS for that task and thus cures the matter. Indeed this works properly for this example, for the TMS produces only the intuitively correct answer. This is caused by its property of possessing *directed* justifications, which excludes the unwanted contrapositives. But this can not be the proper solution to all those problems, because in many cases reasoning in both directions is necessary. Besides, even TMS models like (Giordano & Martelli, 1990*b*) yield both extensions. This seems astonishing, but is simply caused by the fact, that the TMS is not static (this is what is modelled), but changes justifications during DDB. In (Giordano & Martelli, 1990*d*) the Morris example is refuted. In fact *every* existing extension can be reached by TMS, too, if the justifications are supplied to the system in the appropriate order (Giordano & Martelli, 1990*c*; Giordano & Martelli, 1990*b*).

Another good rebuttal of the time argument can be found in (Pequeno, 1990). There it is shown, that time is not the problem. Then the use of *paraconsistency*[12] is recommended.

> "This is the most we can ask from a formalism. We cannot expect it to perform miracles, to extract a single conclusion out of inconclusive knowledge, out of knowledge that supports diverging arguments equally well"

Pequeno forbids the use of contrapositives.

> "... nonmonotonic conclusions cannot have the same epistemological status as conclusions coming from deduction. They can never be dissociated from the evidence that gives support for them. The occurrence of contradiction among them says something about the accuracy of the knowledge available but does not entail the inconsistency of the state of affairs."

Morris (1988*a*) even pleads for an extralogical contradiction handling, which in some sense can be seen parallel to the DDB mechanism in TMS's.

Summarizing, one can say that all of the cures proposed to eliminate the unwanted extensions can be shown as rather artificial, thus solving the concrete problem at hand, but not adequate in general. We hope that our discussion adds some plausibility for the author's strong opinion that the description of the problem does not satisfy the preference for one of the two extensions, though on first view this is suggested. This can be seen very easily when the problem is formulated using our formalism.

[12] Independently invented by da Costa (1974) and Jaskowski (1948); a good survey can be found in (Arruda, 1980).

Chapter 9

Conclusion

We have succeeded in developing a unifying framework for describing reason maintenance systems. This framework captures systems using the justification based approach as well as those that are assumption based, thus making it possible to compare different systems on that basis.

Our approach is purely logical, thus enabling us to

- characterize the differences by naming axioms and translations of the respective elements of a system,
- make apparent the degrees of freedom in the design of such a system.

As has been justified in detail, a combined approach has been chosen, incorporating the reason maintenance system proper as well as the problem solving component. It nevertheless reflects the possibility of modularization, since the two parts appear neatly separated as two distinct parts of the labelled formulae. What is gained, however, is the availability of a description of the interface, a matter very much neglected in past proposals.

9.1 Summary of the Main Results

9.1.1 A Unifying Semantics for Reason Maintenance Systems

Our approach is not restricted to any particular logic. By changing the basic logic, we can dispose of the tight restrictions of existing systems, like the choice of Horn logic or propositional logic. So in fact any arbitrary problem solver, if described in logics at all, can be included. If the component logics possess a model theoretic semantics, then we are able to supply one for the combined system.

The different entailment relations introduced for labelled logics in chapter 4 provide us with a variety of possibilities for system design:

- Simple LL-entailment gives us systems not bound to come up with the "best" result, but capable of deriving intermediate results.

- The introduction of the maximality criterion will enforce a behaviour like the one found in most existing systems: The system gives an answer only if it comes to a quiescent optimal state.
- The consistency demand yields the introduction of nogood handling.
- All combinations are possible, as well as using the approximative entailment relations also presented in chapter 4.

9.1.2 Incremental Calculi

Having started from purely theoretical considerations, we have come to practical systems as well. The abstract reasoner approach has been presented as a way of procedure usable for question answering components — like they appear in expert systems or in user/agent models — particularly interesting in cases when there is no decision procedure available for the basic logic in question and partial answers have to be generated. This fits into the "anytime algorithm" paradigm, which is increasingly recognized as important in AI nowadays.

9.1.3 Handling of Default Reasoning

We have shown that default logics can also be dealt with using our framework originally intended for Reason Maintenance Systems. This is done by changing the labels' logic from propositional logic to first order. It turned out that there is a close resemblance to the abductive approach of Poole.

9.2 Further Work

Starting from the results presented in this thesis there are some potential candidates for further investigation. In the sequel we sketch some of them and rate the value to be expected from pursuing them.

9.2.1 Checking Proof Strategies

In the design of incremental calculi efficiency certainly plays a rôle. In order to achieve this one has to spend some effort on the design of proof strategies. We have already discussed a selection of proof strategies in section 7.2, but it is certainly worthwhile investigating others and check whether they are compatible with our approach.

9.2.2 Modelling Other Systems

What could also be done, is to explicitly prove the correspondence of a model gained from specializing our approach to a particular reason maintenance system implementation. But in our opinion this is not very interesting, because we do not think there could be valuable insights gained from this, and in principle this should not be difficult (though probably tedious), for all the systems fall into one of the main categories dealt with here and the respective differences actually are implementation issues not concerning the logical behaviour.

9.2.3 Varying the Labels' Logic

A particularly interesting point is to examine what can be obtained by varying the logic of the *label part*. This concerns the conjunction and disjunction operators on labels as well as the top level ":" connective, which represents an implication. If we interpret the colon differently from material implication, we can perhaps directly model phenomena like the asymmetry of TMS justifications in a more direct way than our procedure in chapter 5. Also the characterization of dependency directed backtracking may be more elegant than the procedure of section 5.2.1 if a three-valued logic is used for the labels.

All this should in principle work without problems, for as semantics we can always use the very general approach via fibering (cf. section 4.9).

9.2.4 Orderings on Labels

In section 7.5 we have proposed a system that answers queries for F by finding labels for F as well as for $\neg F$. If for both labels are found, we must somehow decide what to give as an answer. One possibility is to confront the user with the (uninterpreted) labels. Another is to introduce a (partial) ordering on labels that at least in some cases enables the system to make the decision.

Such an ordering is a generalization of the ordering on labels we introduced in chapter 4. In default logic terminology this corresponds to *preferences on defaults*, a topic certainly of interest. The algebras necessary for handling this should therefore be explored.

9.2.5 Probabilistic Calculi

A special case, where the introduction of alternative label logics could prove useful, appears in the context of incrementally working calculi for probabilistic or possibilistic logics:

The most prominent possibilistic system is described in (Dubois, Lang & Prade, 1990*b*) (or more detailed in (Dubois, Lang & Prade, 1990*a*)). The approach is *incremental* in nature, using refutation procedures working with resolution. Like ourselves they search for

> "the exploration of all proof paths leading to the empty clause."

In order to overcome combinatoric explosion (labels by far too large given numerous assumptions), the weaker contexts are omitted. They stress that

> "the precise values of certainty degrees are not as important as their ordering".

So why then involve numbers at all. In the author's opinion it is by far more honest (and avoids misleading artifacts) to at best impose a (partial) order on the labels (see previous section).

A system working with genuine probabilities (in contrast to e.g. possibility values) can not be based solely on local inference rules, but must construct complete statistical models. Purely local approaches are therefore *incorrect*, if not particular assumptions of statistical independence are met. These are in fact very often assumed, but unfortunately do very seldom hold in practical applications. If instead some extremely cautious rules are used, the results are not very interesting.

Here an incremental approach with local propagation rules that explicitly account for what has been taken into consideration so far, yields a considerable improvement.

Bibliography

Abiteboul, S. & Grahne, G. (1985), Update semantics for incomplete databases, *in* 'Proceedings of the VLDB Conference', Stockholm.

Anderson, A. & Belnap, N. (1975), *Entailment: The Logic of Relevance and Necessity*, Princeton University Press, Princeton, NJ.

Anderson, R. & Bledsoe, W. W. (1970), 'A linear format for resolution with merging and a new technique for establishing completeness', *Journal of the ACM* **17**, 523–534.

Arruda, A. I. (1980), A survey of paraconsistent logic, *in* A. I. Arruda, R. Chuaqui & N. C. A. da Costa, eds, 'Mathematical Logic in Latin America', North-Holland, Amsterdam.

Baader, F. & Hollunder, B. (1992), Embedding defaults into terminological knowledge representation formalisms, *in* B. Nebel, C. Rich & W. Swartout, eds, 'Principles of Knowledge Representation and Reasoning — Proceedings of the 3rd International Conference', Morgan Kaufmann, Cambridge, Massachusetts, pp. 306–317.

Bibel, W. (1985), Methods of Automated Reasoning, *in* Bibel & Jorrand, eds, 'Fundamentals in Artificial Intelligence', Vol. 232 of *Lecture Notes in Computer Science*, Springer.

Birkhoff, G. & Bartee, T. C. (1970), *Modern Applied Algebra*, Mc Graw-Hill, New York.

Bobrow, D. G., ed. (1980), *Special issue on nonmonotonic logic*, Vol. 13 of *Artificial Intelligence*.

Bossu, G. & Siegel, P. (1985), 'Saturation, nonmonotonic reasoning, and the closed-world assumption', *Artificial Intelligence* **25**, 13–63.

Brewka, G. (1989), Preferred subtheories: An extended logical framework for default reasoning, *in* 'IJCAI-89, Eleventh International Joint Conference on Artificial Intelligence, 20–25 August 1989, Proceedings', Vol. 2, Detroit, Michigan, pp. 1043–1048.

Brewka, G. (1990), On minimal change: A critique of the architecture of nonmonotonic TMS, *in* G. Brewka & U. Junker, eds, 'Aspects of Non-Monotonic Reasoning', GMD, St. Augustin.

Brown, A. L. & Shoham, Y. (1988), New results on semantical nonmonotonic reasoning, *in* 'Proceedings Second International Workshop on Nonmonotonic Reasoning', Vol. 346 of *Lecture Notes in Artificial Intelligence*, Springer, New York, pp. 19–26.

Brown, Jr., A. L. (1985), Modal propositional semantics for reason maintenance systems, *in* 'IJCAI 85, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 18–23 August 1985', Vol. 1, Los Angeles, California, pp. 178–184.

Brown, Jr., A. L. (1988), Logics of justified belief, *in* 'Proceedings of ECAI 88', pp. 507–512.

Brown, Jr., A. L., Gaucas, D. E. & Benanav, D. (1987), An algebraic foundation for truth maintenance, *in* 'IJCAI 87, Proceedings of the Tenth International Joint Conference on Artificial Intelligence, August 23–28, 1987', Vol. 2, Milan, Italy, pp. 973–980.

Cadoli, M. & Schaerf, M. (1991), Approximate entailment, *in* 'Trends in Artificial Intelligence: Proceedings of the 2nd Conference of the Italian Association for Artificial Intelligence', Vol. 549 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 68–77.

Cadoli, M. & Schaerf, M. (1992), Approximate inference in default logic and circumscription, *in* B. Neumann, ed., 'ECAI 92, 10th European Conference on Artificial Intelligence, August 3–7, 1992, Proceedings', Vienna, Austria, pp. 319–323.

Cadoli, M. & Schaerf, M. (n.d.), Approximate inference in default logic and circumscription, version as submitted (to ECAI 92).

Cayrol, M. & Tayrac, P. (1989), ARC: un ATMS basé sur la résolution CAT-correcte, *in* 'Revue d'Intelligence Artificielle', Vol. 3, Hermès, Paris, pp. 19–39.

Clark, K. L. (1978), Negation as failure, *in* H. Gallaire & J. Minker, eds, 'Logic and Data Bases', Plenum Press, New York, pp. 293–322.

da Costa, N. C. A. (1974), 'On the theory of inconsistent formal systems', *Notre Dame Journal of Formal Logic*.

de Kleer, J. (1984), Choices without backtracking, *in* 'Proceedings of the National Conference on Artificial Intelligence, August 6–10, 1984, AAAI-84', University of Texas at Austin, pp. 79–85.

de Kleer, J. (1986*a*), 'An assumption–based TMS', *Artificial Intelligence* **28**(1), 127–162.

de Kleer, J. (1986*b*), 'Extending the ATMS', *Artificial Intelligence* **28**(1), 163–196.

de Kleer, J. (1986*c*), 'Problem solving with the ATMS', *Artificial Intelligence* **28**(1), 197–224.

de Kleer, J. (1989), A comparison of ATMS and CSP techniques, *in* 'IJCAI-89, Eleventh International Joint Conference on Artificial Intelligence, 20–25 August 1989, Proceedings', Vol. 1, Detroit, Michigan, pp. 290–296.

de Kleer, J. & Williams, B. C. (1986*a*), Back to backtracking: Controlling the ATMS, *in* 'Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, August 11-15, 1986', Vol. 2, Philadelphia, PA, pp. 910–917.

de Kleer, J. & Williams, B. C. (1986*b*), Reasoning about multiple faults, *in* 'Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, August 11-15, 1986', Vol. 1, Philadelphia, PA, pp. 132–139.

de Kleer, J. & Williams, B. C. (1987), 'Diagnosing multiple faults', *Artificial Intelligence* **32**(2), 97–130.

Dean, T. & Boddy, M. (1988), An analysis of time-dependent planning, *in* 'AAAI–88, the Seventh National Conference on Artificial Intelligence', Saint Paul, Minnesota, pp. 49–54.

Demolombe, R. & Fariñas del Cerro, L. (1991), An inference rule for hypothesis generation, *in* 'IJCAI-91,12th International Joint Conference on Artificial Intelligence, 24–30 August 1991', Vol. 1, Darling Harbour, Sydney, Australia, pp. 152–157.

Doyle, J. (1979), 'A truth maintenance system', *Artificial Intelligence* **12**, 231–272.

Doyle, J. (1983), The ins and outs of reason maintenance, *in* 'IJCAI-83, Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 8–12 August 1983', Vol. 1, Karlsruhe, West Germany, pp. 349–351.

Dressler, O. (1989), An extended basic ATMS, *in* 'Proceedings of the Second International Workshop on Non-Monotonic Reasoning', Vol. 343 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 143–163.

Dressler, O. & Farquhar, A. (1990), Putting the problem solver back in the driver's seat: Contextual control of the ATMS, *in* J. P. Martins & M. Reinfrank, eds, 'Truth Maintenance Systems', Vol. 515 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 1–16.

Dubois, D., Lang, J. & Prade, H. (1990*a*), Handling uncertain knowledge in an ATMS using possibilistic logic, a revised version of this draft will appear in the Proceedings of the Fifth International Symposium on Methodologies for Intelligent Systems.

Dubois, D., Lang, J. & Prade, H. (1990*b*), A possibilistic assumption-based truth maintenance system with uncertain justifications, and its application to belief revision, *in* J. P. Martins & M. Reinfrank, eds, 'Truth Maintenance

Systems', Vol. 515 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 87–106.

Dunn, J. M. (1986), Relevance logic and entailment, *in* D. Gabbay & F. Guenthner, eds, 'Handbook of Philosophical Logic', Vol. III, Reidel Publishing Company, chapter III.3, pp. 117–224.

Eiter, T. & Gottlob, G. (1991), Propositional circumscription and extended closed world reasoning are Π_2^p-complete, Technical Report CD-TR 91/20, Technische Universität Wien, Christian Doppler Labor für Expertensysteme, Wien.

Elkan, C. (1989), Logical characterizations of nonmonotonic TMSs, *in* A. Kreczmar & G. Mirkowska, eds, 'Mathematical Foundations of Computer Science 1989', Springer, Heidelberg, pp. 218–224.

Elkan, C. (1990), 'A rational reconstruction of nonmonotonic truth maintenance systems', *Artificial Intelligence* **43**(2), 219–234.

Eshghi, K. & Kowalski, R. A. (1988), Abduction as deduction, Technical report, Dept. of Computing, Imperial College of Science and Technology, London.

Eshghi, K. & Kowalski, R. A. (1989), Abduction compared with negation by failure, *in* G. Levi & M. Martelli, eds, 'Proceedings of the ICLP89', IEEE, MIT Press, Lisbon, Portugal, pp. 234–254.

Euzenat, J. (1992), A potentially complete algorithm for non monotonic reason maintenance, submitted to ECAI 92.

Fagin, R., Kuper, G. M., Ullman, J. D. & Vardi, M. Y. (1984), Updating logical databases, *in* 'Proceedings of the 3rd ACM PODS'.

Fagin, R., Ullman, J. D. & Vardi, M. Y. (1983), On the semantics of updates in databases, *in* 'Proceedings of the 2nd ACM PODS'.

Fehrer, D. (1993), A Unifying Framework for Reason Maintenance, *in* M. Clarke, R. Kruse & S. Moral, eds, 'Symbolic and Quantitative Approaches to Reasoning and Uncertainty, Proceedings of ECSQARU '93, Granada, Spain, Nov. 1993', Vol. 747 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 113–120.

Fehrer, D. (1994), A unifying logical framework for reason maintenance (deliverable DI.1.2-3P), *in* J. Cunningham & J. Pitt, eds, 'Medlar II Report PPR2', Vol. 1, Department of Computing, Imperial College of Science, Technology & Medicine, pp. 47–55.

Fehrer, D., Hustadt, U., Jaeger, M., Nonnengart, A., Ohlbach, H. J., Schmidt, R. A., Weidenbach, C. & Weydert, E. (1994), Description logics for natural language processing, *in* F. Baader, M. Lenzerini, W. Nutt & P. F. Patel-Schneider, eds, 'Working Notes of the 1994 Description Logic Workshop', pp. 80–84.

Frege, G. (1879), *Begriffsschrift, eine der Arithmetischen nachgebildete Formelsprache des reinen Denkens*, Verlag von Louis Nebert, Halle.

Frost, R. A. (1986), *Introduction to Knowledge Base Systems*, Collins, London.

Fujiwara, Y. & Honiden, S. (1989), Relating the TMS to autoepistemic logic, *in* 'IJCAI-89, Eleventh International Joint Conference on Artificial Intelligence, 20–25 August 1989, Proceedings', Vol. 2, IJCAI-89, Eleventh International Joint Conference on Artificial Intelligence, 20–25 August 1989, Proceedings, pp. 1199–1205.

Fujiwara, Y. & Honiden, S. (1990), On logical foundations of the ATMS, *in* J. P. Martins & M. Reinfrank, eds, 'Truth Maintenance Systems', Vol. 515 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 125–135.

Gabbay, D. M. (1985), Theoretical foundations for nonmonotonic reasoning in expert systems, *in* K. R. Apt, ed., 'Proceedings NATO Advanced Study Institute on Logics and Models of Concurrent Systems', La Colle-sur-Loup, France, Springer Verlag, Berlin, pp. 439–457.

Gabbay, D. M. (1991), Labelled deductive systems, part I, CIS Bericht 90-22, Centrum für Informations- und Sprachverarbeitung.

Gabbay, D. M. (1994*a*), Combining labelled deductive systems (deliverable DIII.1.2–1P), *in* J. Cunningham & J. Pitt, eds, 'Medlar II Report PPR2', Vol. 2, Department of Computing, Imperial College of Science, Technology & Medicine, pp. 285–334.

Gabbay, D. M. (1994*b*), LDS — labelled deductive systems, volume 1 — foundations, MPI-report MPI–I–94–223, Max-Planck-Institut für Informatik, Saarbrücken.

Gärdenfors, P. (1988), *Knowledge in Flux — Modeling the Dynamics of Epistemic States*, The MIT Press, Cambridge, Mass.

Gärdenfors, P. (1990), Belief revision and nonmonotonic logic: two sides of the same coin?, *in* L. Carlucci Aiello, ed., 'ECAI 90, 9th European Conference on Artificial Intelligence, August 6–10, 1990, Proceedings', Stockholm, Sweden, pp. 768–773.

Gelfond, M. & Lifschitz, V. (1988), The stable model semantics for logic programming, *in* K. Bowen & R. Kowalski, eds, 'Fifth International Conference Symposium on Logic Programming, Seattle, WA', Vol. 2, MIT Press, Cambridge, MA, pp. 1070–1080.

Giordano, L. & Martelli, A. (1990*a*), An Abductive Characterization of the TMS, *in* L. Carlucci Aiello, ed., 'ECAI 90, 9th European Conference on Artificial Intelligence, August 6–10, 1990, Proceedings', Stockholm, Sweden, pp. 308–313.

Giordano, L. & Martelli, A. (1990*b*), Generalized stable models, truth maintenance and conflict resolution, *in* D. H. D. Warren & P. Szeredi, eds, 'Logic Programming. Proceedings of the Seventh International Conference', Logic Programming Series, MIT Press, Cambridge, Massachusetts / London, England, pp. 427–441.

Giordano, L. & Martelli, A. (1990*c*), Three-valued labellings for truth maintenance systems, *in* 'Proceedings ISMIS', Knoxville, pp. 506–513.

Giordano, L. & Martelli, A. (1990*d*), Truth maintenance systems and belief revision, *in* J. P. Martins & M. Reinfrank, eds, 'Truth Maintenance Systems', Vol. 515 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 71–86.

Goodwin, J. W. (1982), An improved algorithm for non–monotonic dependency net update, Technical Report LITH-MAT-R-82-83, Linköping University.

Goodwin, J. W. (1987), A Theory and System for Non-Monotonic Reasoning, Linköping studies in science and technology, dissertations, no.165, Department of Computer and Information Science, Linköping University, S-581 83 Linköping.

Gottlob, G. (1991), Complexity results for nonmonotonic logics, Technical Report CD-TR 91/24, Technische Universität Wien, Christian Doppler Labor für Expertensysteme, Wien.

Haddawy, P. & Frisch, A. M. (1991), Anytime deduction for probabilistic logic, Technical report, University of Illinois. also to appear in Artificial Intelligence, autumn 1994.

Haneclou, P. (1987), A formal approach to reason–maintenance based on abstract domains, Technical Report LITH-IDA-R-87-07, Institutionen för datavetenskap, Universitetet i Linköping och Tekniska Högskolan, S-581 83 Linköping.

Hanks, S. & McDermott, D. V. (1985), Temporal reasoning and default logics, Technical Report YALU/CSD/RR 430, Yale University.

Hanks, S. & McDermott, D. V. (1986), Default reasoning, nonmonotonic logics, and the frame problem, *in* 'Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, August 11-15, 1986', Vol. 1, Philadelphia, PA, pp. 328–333.

Hanks, S. & McDermott, D. V. (1987), 'Nonmonotonic logic and temporal projection', *Artificial Intelligence* **33**, 374–412.

Horvitz, E. (1989), Reasoning about beliefs and actions under computational resource constraints, *in* L. Kanal, T. Levitt & J. Lemmer, eds, 'Uncertainty in Artificial Intelligence', Vol. 3, Elsevier, pp. 301–324.

Imielinski, T. & Lipski, W. (1984), 'Incomplete information in relational databases', *Journal of the ACM*.

Inoue, K. (1990), An abductive procedure for the CMS/ATMS, *in* J. P. Martins & M. Reinfrank, eds, 'Truth Maintenance Systems', Vol. 515 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 34–53.

Inoue, K. (1992), 'Linear resolution for consequence finding', *Artificial Intelligence* **56**, 301–353.

Israel, D. J. (1980), What's wrong with non-monotonic logic?, *in* 'Proceedings of the First Annual National Conference on Artificial Intelligence at Stanford University, August 18 to 21, 1980', pp. 99–101.

Jaskowski, S. (1948), 'Un calcul des propositions pour les systèmes deductifs contradictoires', Studia Soc. Scien. Torunensis, Section A, 1.

Junker, U. (1987), Diskussion und Implementierung eines annahmenbasierten Truth Maintenance Systems, Technical report, GMD.

Junker, U. (1989), A correct non-monotonic ATMS, *in* 'IJCAI-89, Eleventh International Joint Conference on Artificial Intelligence, 20–25 August 1989, Proceedings', Detroit, Michigan.

Junker, U. (1990), Variations on backtracking for TMS, *in* J. P. Martins & M. Reinfrank, eds, 'Truth Maintenance Systems', Vol. 515 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 17–33.

Junker, U. & Konolige, K. (1990*a*), Computing the extensions of autoepistemic and default logics with a truth maintenance system, to appear in AAAI 90.

Junker, U. & Konolige, K. (1990*b*), Computing the extensions of autoepistemic and default logics with a truth maintenance system, *in* 'AAAI-90, Proceedings Eighth National Conference on Artificial Intelligence, July 29 – Aug. 3, 1990', Vol. 1, AAAI Press/ The MIT Press, pp. 278–283.

Kakas, A. C. & Mancarella, P. (1990*a*), Generalized stable models: a semantics for abduction, *in* L. Carlucci Aiello, ed., 'ECAI 90, 9th European Conference on Artificial Intelligence, August 6–10, 1990, Proceedings', Stockholm, Sweden, pp. 385–391.

Kakas, A. C. & Mancarella, P. (1990*b*), Knowledge assimilation and abduction, *in* J. P. Martins & M. Reinfrank, eds, 'Truth Maintenance Systems', Vol. 515 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 54–70.

Katsuno, H. & Satoh, K. (1991), A unified view of consequence relation, belief revision and conditional logic, *in* 'IJCAI-91,12th International Joint Conference on Artificial Intelligence, 24–30 August 1991', Vol. 1, Darling Harbour, Sydney, Australia, pp. 406–412.

Kautz, H. A. (1986), The logic of persistence, *in* 'Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, August 11-15, 1986', Vol. 1, Philadelphia, PA, pp. 401–405.

Kautz, H. A. & Selman, B. (1991), 'Hard problems for simple default logics', *Artificial Intelligence* **49**, 243–279.

Kean, A. & Tsiknis, G. (1988), An incremental method for generating prime implicants/implicates, Technical Report 88–16, Department of Computer Science, The University of British Columbia.

Konolige, K. (1983), A deductive model of belief, *in* 'IJCAI-83, Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 8–12 August 1983', Vol. 1, Karlsruhe, West Germany, pp. 377–381.

Konolige, K. (1988), 'On the relation between default and autoepistemic logic', *Artificial Intelligence* **35**, 343–382.

Konolige, K. (1989), On the relation between autoepistemic logic and circumscription, *in* 'IJCAI-89, Eleventh International Joint Conference on Artificial Intelligence, 20–25 August 1989, Proceedings', Vol. 2, Detroit, Michigan, pp. 1213–1218.

Kraus, S., Lehmann, D. & Magidor, M. (1990), 'Nonmonotonic reasoning, preferential models and cumulative logics', *Artificial Intelligence* **44**(1–2), 167–207.

Lee, R. C. T. (1967), A completeness theorem and a computer program for finding theorems derivable from given axioms, PhD thesis, University of California, Berkeley.

Levesque, H. J. (1984*a*), 'Foundations of a functional approach to knowledge representation', *Artificial Intelligence* **23**, 155–212.

Levesque, H. J. (1984*b*), The logic of incomplete knowledge bases, *in* M. L. Brodie, J. Mylopoulos & J. W. Schmidt, eds, 'On Conceptual Modelling', Springer.

Levesque, H. J. (1989), A knowledge–level account of abduction, *in* 'IJCAI-89, Eleventh International Joint Conference on Artificial Intelligence, 20–25 August 1989, Proceedings', Vol. 2, Detroit, Michigan, pp. 1061–1067.

Lifschitz, V. (1985), Computing circumscription, *in* 'IJCAI 85, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 18–23 August 1985', Vol. 1, Los Angeles, California, pp. 121–127.

Lifschitz, V. (1986), Pointwise circumscription: Preliminary report, *in* 'Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, August 11-15, 1986', Vol. 1, Philadelphia, PA, pp. 406–410.

Lifschitz, V. (1987), Pointwise circumscription, *in* M. L. Ginsberg, ed., 'Readings in Nonmonotonic Logic', Morgan Kaufmann Publishers, Inc., Los Altos, California, chapter 3, pp. 179–193. Stanford preprint.

Makinson, D. (1989), General theory of cumulative inference, *in* M. Reinfrank, J. de Kleer, M. L. Ginsberg & E. Sandewall, eds, 'Non-Monotonic Reasoning (2nd Intern. Workshop, Grassau, FRG, June 1988)', Vol. 346 of *Lecture Notes in Computer Science*, Springer Verlag, Berlin, pp. 1–18.

Marek, W. & Truszczyński, M. (1989), Relating autoepistemic and default logics, *in* 'Principles of Knowledge Representation and Reasoning', Morgan Kaufmann, San Mateo, CA, pp. 276–288.

Marek, W. & Truszczyński, M. (1990), 'Modal logic for default reasoning', *Annals of Mathematics and Artificial Intelligence* **1**, 275–302.

Martins, J. P. (1983), Reasoning in Multiple Belief Spaces, Computer science technical report no. 203, Department of Computer Science, State University of New York, Buffalo, NY.

Martins, J. P. & Shapiro, S. C. (1983), Reasoning in multiple belief spaces, *in* 'IJCAI-83, Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 8–12 August 1983', Vol. 1, Karlsruhe, West Germany, pp. 370–373.

McAllester, D. (1978), A three valued truth maintenance system, AI Memo 473, AI lab, MIT, Cambridge, MA.

McAllester, D. (1980), An outlook on truth maintenance, AI Memo AIM-551, Artificial Intelligence Laboratory, M.I.T., Cambridge, MA.

McAllester, D. (1982), Reasoning utility package user's manual, AI Lab Report AIM-667, Artificial Intelligence Laboratory, M.I.T., Cambridge, MA.

McCarthy, J. (1980), 'Circumscription — a form of non-monotonic reasoning', *Artificial Intelligence* **13**, 27–39.

McCarthy, J. (1984), Applications of circumscription to formalizing common sense knowledge, *in* 'Proceedings of the Non-Monotonic Reasoning Workshop, AAAI, Oct. 17–19', New Paltz, NY, pp. 295–324.

McCarthy, J. & Hayes, P. J. (1969), Some philosophical problems from the standpoint of artificial intelligence, *in* B. Meltzer & D. Michie, eds, 'Machine Intelligence', Vol. 4, Edinburgh University Press, chapter 26, pp. 463–502.

McDermott, D. V. (1974), Assimilation of new information by a natural language understanding system, Mit artificial intelligence laboratory technical report 291, MIT Department of Electrical Engineering and Computer Science.

McDermott, D. V. (1980), Non-monotonic logic II, Research Report 174, Dept. of Computer Science, Yale University, New Haven, Conn.

McDermott, D. V. (1982*a*), 'Nonmonotonic logic II: Nonmonotonic modal theories', *Journal of the ACM* **29**(1), 33–57.

McDermott, D. V. (1982*b*), 'A temporal logic for reasoning about processes and plans', *Cognitive Science* **6**, 101–155.

McDermott, D. V. (1983), 'Contexts and data dependencies: a synthesis', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **5**(3), 237–246.

McDermott, D. V. (1991), 'A general framework for reason maintenance', *Artificial Intelligence* **50**(3), 289–329.

McDermott, D. V. & Doyle, J. (1980), 'Non-monotonic logic I', *Artificial Intelligence* **13**, 41–72.

Minicozzi, E. & Reiter, R. (1972), 'A note on linear resolution strategies in consequence finding', *Artificial Intelligence* **3**, 175–180.

Moore, R. C. (1983), Semantical considerations on nonmonotonic logic, *in* 'IJCAI-83, Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 8–12 August 1983', Karlsruhe, West Germany, pp. 272–279.

Moore, R. C. (1985), 'Semantical considerations on nonmonotonic logic', *Artificial Intelligence* **25**, 75–94.

Morris, P. H. (1987), Curing anomalous extensions, *in* 'AAAI 87, July 13–17, 1987, Sixth National Conference on Artificial Intelligence', Vol. 2, Seattle, Washington, pp. 437–442.

Morris, P. H. (1988*a*), 'The anomalous extension problem in default reasoning', *Artificial Intelligence* **35**(2), 383–399.

Morris, P. H. (1988*b*), Autoepistemic stable closures and contradiction resolution, *in* 'Non-monotonic Reasoning, 2nd International Workshop', Vol. 346 of *Lecture Notes in Computer Science*, Springer Verlag.

Morris, P. H. (1988*c*), Stable closures, defeasible logic, and contradiction tolerant reasoning, *in* 'AAAI 88, The Seventh National Conference on Artificial Intelligence, Aug. 21–26, 1988', Vol. 2, Saint Paul, Minnesota, pp. 506–511.

Nicolas, J. M. & Yazdanian, K. (1978), Integrity checking indeductive databases, *in* H. Gallaire & J. Minker, eds, 'Logic and Data Bases', Plenum Press, New York.

Pequeno, T. (1990), A logic for inconsistent nonmonotonic reasoning, Imperial College Research Report DoC 90/6, Department of Computing, Imperial College of Science, Technology and Medicine, 189 Queen's Gate, London SW7 2BZ.

Petrie, Jr., C. J. (1987), Revised dependency-directed backtracking for default reasoning, *in* 'AAAI 87, July 13–17, 1987, Sixth National Conference on Artificial Intelligence', Vol. 1, Seattle, Washington, pp. 167–172.

Pimentel, S. G. & Cuadrado, J. L. (1989), A truth maintenance system based on stable models, *in* 'Proceedings North American Conference on Logic Programming', Cleveland, pp. 274–290.

Poole, D. L. (1984), A computational logic of default reasoning, Computer science research report, University of Waterloo.

Poole, D. L. (1985), On the comparison of theories: Preferring the most specific explanation, *in* 'IJCAI 85, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 18–23 August 1985', Vol. 1, Los Angeles, California, pp. 144–147.

Poole, D. L. (1988*a*), 'A logical framework for default reasoning', *Artificial Intelligence* **36**(3), 27–47.

Poole, D. L. (1988*b*), A methodology for using a default and abductive reasoning system, Technical report, Dept. of Computer Science, University of Waterloo, Waterloo, Ont.

Pratt, I. (1994), *Artificial Intelligence*, Macmillan, London.

Reinfrank, M. (1989), Fundamentals and logical foundations of truth maintenance, Linköping studies in science and technology 221, Linköping University.

Reinfrank, M., Dressler, O. & Brewka, G. (1989), On the relation between truth maintenance and autoepistemic logic, *in* 'IJCAI-89, Eleventh International Joint Conference on Artificial Intelligence, 20–25 August 1989, Proceedings', Vol. 2, Detroit, Michigan, pp. 1206–1212.

Reiter, R. (1980*a*), Data bases: a logical perspective, *in* 'ACM SIGART, SIGMOD, SIGPLAN Proceedings Workshop on Data Abstraction, Data Bases and Conceptual Modelling', Pingree Park, Colorado.

Reiter, R. (1980*b*), 'A logic for default reasoning', *Artificial Intelligence* **13**(1), 81–132.

Reiter, R. (1984), Towards a logical reconstruction of relational database theory, *in* M. Brodie, J. Myopoulos & J. Schmidt, eds, 'On Conceptual Modelling', Springer Verlag.

Reiter, R. (1987), 'A theory of diagnosis from first principles', *Artificial Intelligence* **32**(2), 57–95.

Reiter, R. & Criscuolo, G. (1981), On interacting defaults, *in* 'Proceedings of the Seventh International Joint Conference on Artificial Intelligence, IJCAI-81, 24–28 August 1981', Vol. 1, University of British Columbia, Vancouver, B.C., Canada, pp. 270–276.

Reiter, R. & de Kleer, J. (1987), Foundations of assumption–based truth maintenance systems: Preliminary report, *in* 'AAAI 87, July 13–17, 1987, Sixth

National Conference on Artificial Intelligence', Vol. 1, Seattle, Washington, pp. 183–188.

Rescher, N. (1964), *Hypothetical Reasoning*, North-Holland Publ., Amsterdam.

Robinson, J. A. (1965), 'A machine–oriented logic based on the resolution principle', *Journal of the ACM* **12**(1), 23–41.

Robinson, J. A. (1967), A review of automated theorem proving, *in* 'Proceedings of the Symposium in Applied Mathematics', Amer. Math. Soc., Providence, R. I.

Routley, R. & Meyer, R. K. (1973), The semantics of entailment I, *in* Leblanc, ed., 'Truth, Syntax and Modality', North Holland, pp. 199–243.

Russell, S. J. & Zilberstein, S. (1991), Composing real-time systems, *in* 'IJCAI–91, Proceedings of the 12th International Conference on Artificial Intelligence', Darling Harbour, Sydney, pp. 212–217.

Shoham, Y. (1986), Chronological ignorance; time, nonmonotonicity and causal theories, *in* 'Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, August 11-15, 1986', Vol. 1, Philadelphia, PA, pp. 389–393.

Shoham, Y. (1988), 'Chronological ignorance: Experiments in nonmonotonic temporal reasoning', *Artificial Intelligence* **36**, 279–331.

Shortliffe, E. H. (1976), *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York.

Slagle, J. R. (1967), 'Automated theorem proving with renamable and semantic resolution', *Journal of the ACM* **12**(4), 687–697.

Slagle, J. R., Chang, C.-L. & Lee, R. C. T. (1969), Completeness theorems for semantic resolution in consequence–finding, *in* 'Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-69, 7–9 May, 1969', Washington, D.C., pp. 281–285.

Sombé, L. (1990), 'Reasoning under incomplete information in artificial intelligence: A comparison of formalisms using a single example', *Int. J. of Intelligent Systems* **5**(4), 323–471. available also as a monograph, Wiley, New York.

Stallman, R. M. & Sussman, G. J. (1977), 'Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis', *Artificial Intelligence* **9**, 135–196.

Steele, G. (1979), The definition and implementation of a computer programming language based on constraints, Technical Report TR 595, Artificial Intelligence Laboratory, M.I.T., Cambridge.

Stillman, J. (1990), It's not my default: The complexity of membership problems in restricted propositional default logics, *in* 'AAAI-90, Proceedings Eighth National Conference on Artificial Intelligence, July 29 – Aug. 3, 1990', Vol. 2, AAAI Press/ The MIT Press, pp. 571–578.

Tayrac, P. (1990), ARC: An extended ATMS based on directed CAT-correct resolution, *in* J. P. Martins & M. Reinfrank, eds, 'Truth Maintenance Systems', Vol. 515 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 107–123.

Tison, P. (1967), 'Generalized consensus theory and application to the minimization of Boolean functions', *IEEE transactions on electronic computers* **16**, 446–456.

Winslett, M. (1986), Is belief revision harder than you thought?, *in* 'Proceedings AAAI-86, Fifth National Conference on Artificial Intelligence, August 11-15, 1986', Philadelphia, PA, pp. 421–427.

Winslett-Wilkins, M. (1986), A model-theoretic approach to updating logical databases, Technical report, Stanford Computer Science Dept.

Witteveen, C. (1990), A skeptical semantics for truth maintenance, *in* J. P. Martins & M. Reinfrank, eds, 'Truth Maintenance Systems', Vol. 515 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 136–154.

Wittgenstein, L. (1975), *Philosophical Remarks*, Basil Blackwell. the original is from 1930.

Xianchang, W. & Huowang, C. (1991), On semantics of TMS, *in* 'IJCAI-91,12th International Joint Conference on Artificial Intelligence, 24–30 August 1991', Vol. 1, Darling Harbour, Sydney, Australia, pp. 306–309.

Table of Defined Symbols

L .... believe-operator in AEL (91)

M .... consistent-to-assume operator, e.g. in NML I and DL (112)

$\Im_{BL}$ .... interpretation of the basic logic (61)

$\Im_{LL}$ .... interpretation of labelled logic (61)

$\Im_{PL}$ .... interpretation of (classical) propositional logic (61)

Ψ .... used in computing restriction sets (84)

χ .... characteristic function (38), see also (117)

λ .... canonical label (70)

μ .... auxiliary function for χ (38), for restriction sets (84), see also (117)

σ .... used in computing restriction sets (84)

$\widetilde{A}$ .... the corresponding atom to node A (67)

α_A .... the corresponding label to node A (67)

$\models_{\mathrm{BL}}$ .... entails in the basic logic (36)

$\models_{\mathrm{LL}}$ .... LL-entails (37)

$\models_{\mathrm{PL}}$ .... entails in classical propositional logic (36)

$\models_{\mathrm{LL(BL)}}$ .... entails in labelled logic with BL as basic logic (97)

$\models_{\mathrm{consLL}}$ .... consistently LL-entails (52)

$\models_{\mathrm{equivLL}}$ .... strictly LL-entails (36)

$\models_{\mathrm{inputLL}}$ .... input-LL-entails (53)

$\models_{\mathrm{max(E)}}$ .... maximally E-entails (56)

$\models_{\mathrm{max(LL)}}$ .... maximally LL-entails (49)

$\models_{\Omega\text{-}\mathrm{LL}}$ .... Ω-LL-entails (53)

$\models_{\mathrm{str}\text{-}\Omega\text{-}\mathrm{LL}}$ .... strictly Ω-LL-entails (53)

$\models_{\omega\text{-}\mathrm{LL}}$ .... ω-LL-entails (52)

$\varphi \equiv \psi$ .... $\varphi \models_{\mathrm{LL}} \psi$ and $\psi \models_{\mathrm{LL}} \varphi$ (44)

i .... used in computing upper closed sets (90)

l .... labelling function (11), used in computing upper closed sets (90)

$+$ .... disjunction in labels (35)

$\overline{\alpha}$ .... negation of label α (35)

$\sum$ .... disjunction over sets of labels (38)

$\prod$ .... conjunction over sets of labels (38)

$\mathrm{LL}_{\mathrm{ATMS}}(\mathrm{DN})$ .... labelled representation of ATMS DN (66)

$\mathrm{LL}_{\mathrm{TMS}}(\mathrm{DN})$ .... labelled representation of TMS DN (67)

$\mathrm{LL}_{\mathrm{TMS'}}(\mathrm{DN})$ labelled representation of TMS DN with contradiction nodes (68)

LL(BL) .... labelled logic with BL as basic logic (96)

LL(Φ) .... labelled representation of unlabelled formula set Φ (97)

$\vee$ .... logical disjunction (35)

$\wedge$ .... logical conjunction (35)
$\neg$ .... logical negation (35)
$\leftrightarrow$ .... equivalence in PL or BL (41)
$\rightarrow$ .... implication in PL or BL (41)
$\emptyset$ .... the empty set (42)
$\top$ .... the verum (35)
$\perp$ .... contradiction node (17), the falsum (35)
Cons .... the set of logical consequences (42)
conseq .... yields the consequent of a justification (11)
formula .... formula part of a labelled formula (36)
inset .... yields the IN-SET of a justification (11)
label .... label part of a labelled formula (36), inverse of χ (38)
maxlabel(F, Φ) .... the maximal label for F wrt. Φ (49)
outset .... yields the OUT-SET of a justification (11)
rank .... ranking function (14)
ucl .... upper closed sets (91)
`contradiction` .... truth value (91)
`false` .... truth value (91, 92)
`in` .... label for node currently believed (11), truth value (92)
`out` .... label for node currently not believed (11), truth value (92)
`true` .... truth value (91)
`undefined` .... truth value (91)
`undet` .... label for node with unknown belief status (11)

Abbreviations

A

AEL .... autoepistemic logic
AI .... artificial intelligence
AR .... abstract reasoner
ATMS .... (de Kleer's) assumption based truth maintenance system
ATP .. automated theorem proving

B

BL .... basic logic

C

CMS ... clause management system
CNF .... clause normal form
CP . conditional proof (justification)
CR .... contraction rule

D

DDB .... dependency directed backtracking
DDNF .... distinguished disjunctive normal form
DL .... Reiter's default logic
DN .... dependency net
DNF ...... disjunctive normal form

J

JTMS . (Doyle's) justification based truth maintenance system

L

LDS ..... labelled deductive system
LL .... labelled logics

M

MBR ...... multiple belief reasoner

N

NF .... nogood filtration rule
NMFS ...... non monotonic formal system

P

PL .... propositional logic

R

RMS . Reason Maintenance System

S

SL ...... support list (justification)
SNeBR SNePS with Belief Revision
SNePS .... Semantic Network Processing System
SOS .... set of support
SWM . Shapiro, Wand and Martins logic

T

TMP . truth maintenance procedure
TMS .. (Doyle's) truth maintenance system

Index

Y