LABELLED DEDUCTIVE SYSTEMS: A POSITION PAPER

D. M. GABBAY

§1. Labelled deductive systems in context.

The purpose of this paper is to introduce a general notion of a logical system, namely that of a *Labelled Deductive System* (*LDS*), and show that many logical systems, new and old, monotonic and non-monotonic, all fall within this new framework. This research will eventually be published as a book, and this paper is based on Chapter 1 of [19].

We begin with the traditional view of what is a logical system.

Traditionally, to present a logic **L**, we need to present first the set of well-formed formulas of that logic. This is the *language* of the logic. We specify the sets of atomic formulas, connectives, quantifiers and the set of well-formed formulas. Secondly, we mathematically define the notion of consequence, that is, for sets of formulas Δ and formulas Q, we define the consequence relation $\Delta \vdash_{\mathbf{L}} Q$, which is read "$Q$ follows from Δ in the logic **L**".

The consequence relation is required to satisfy the following intuitive properties: (Δ, Δ' abbreviates $\Delta \cup \Delta'$).

Reflexivity

$$\Delta \vdash Q \text{ if } Q \in \Delta$$

Monotonicity

$$\frac{\Delta \vdash Q}{\Delta, \Delta' \vdash Q}$$

1.1. Transitivity (cut)

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

If you think of Δ as a database and Q as a query, then reflexivity means that the answer "yes" is given for any Q which is already listed in the database Δ. Monotonicity reflects the accumulation of data, and transitivity is nothing but lemma generation, namely, if $\Delta \vdash A$, then A can be used as a lemma to derive B from Δ.

These three properties have appeared to constitute minimal and most natural for a logical system, given that the main applications of logic were in mathematics and philosophy.

The above notions were essentially put forward by Tarski [8] in 1936 and is referred to as Tarski consequence. Scott [7], inspired by Gabbay [16], generalised the notion to allow Q to be a set of formulas Γ. The basic relation is then of the form $\Delta \vdash \Gamma$, satisfying:[1]

Reflexivity

$$\Delta \vdash \Gamma \text{ if } \Delta \cap \Gamma \neq \varnothing$$

Monotonicity

$$\frac{\Delta \vdash \Gamma}{\Delta, \Delta' \vdash \Gamma}$$

1.2. Transitivity (cut)

$$\frac{\Delta, A \vdash \Gamma; \Delta' \vdash A, \Gamma'}{\Delta, \Delta' \vdash \Gamma, \Gamma'}$$

Scott further showed that for any Tarski consequence relation $\vdash$ there exist two Scott consequence relations (a maximal one and a minimal one) that agree with it, namely, that $\Delta \vdash A$ (Tarski) iff $\Delta \vdash \{A\}$ (Scott) (see Gabbay [2]).

The above notions are monotonic. However, the increasing use of logic in theoretical computer science and artificial intelligence has given rise to logical systems which are not monotonic, i.e., to systems in which the axiom of monotonicity is not satisfied. There are many such systems, satisfying a variety of conditions and presented in a variety of ways. Furthermore, some are characterized in a proof-theoretical and some in a model-theoretical manner. All these different presentations give rise to some notion of consequence $\Delta \vdash Q$, but they only seem to all agree on reflexivity. The essential difference between these logics (commonly called *non-monotonic logics*) and the more traditional logics (now referred to as *monotonic logics*) is the fact that $\Delta \vdash A$ holds in the monotonic case because of some $\Delta_A \subseteq \Delta$, while in the non-monotonic case the entire set Δ is somehow used to derive A. Thus if Δ is increased to Δ', there is no change in the monotonic case, while there may be a change in the non-monotonic case.

The above describes the situation current in the early 1980's. We have had a multitude of systems generally accepted as "logics" without a unifying underlying theory and many had semantics without proof theory or vice-versa, though almost all of them were based on some sound intuitions of one form or another. Clearly there was the need for a general unifying framework. An early attempt at classifying non-monotonic systems was Gabbay [3]. It was put forward that basic axioms for a Tarski type consequence relation should be *reflexivity, cut* (version 1.2 above) and *restricted monotonicity*, namely:

Restricted monotonicity (cumulativity)

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

[1]The similarity with Gentzen sequents is obvious. A sequent $\Delta \vdash \Gamma$ is a relation between Δ and Γ. Such a relation can either be defined axiomatically (as a consequence relation) or be generated via closure conditions like $A \vdash A$ (initial) and other generating rules. The generating rules correspond to Gentzen rules. In many logics we have $\Delta \vdash \Gamma$ iff $\emptyset \vdash \bigwedge \Delta \rightarrow \bigvee \Gamma$, which gives an intuitive meaning to $\vdash$.

A variety of systems seem to satisfy this axiom. Further results were obtained (Lehmann [11, 12]), (Wojcicki [9, 10]), (Makinson [5, 6]) and the area was called "axiomatic theory of the consequence relation" by Wojcicki. A recent general theory is presented in Gabbay [20, 21].

Although some sort of classification was obtained and semantical results were proved, the approach does not seem to be strong enough. Many systems do not satisfy restricted monotonicity. Other systems such as relevance logic, do not even satisfy reflexivity. Others have a richness of their own which is lost in a simple presentation as an axiomatic consequence relation. Obviously a different approach is needed, one which would be more sensitive to the variety of features of the systems in the field. Fortunately, developments in a neighbouring area, that of automated deduction, seem to be of help. New automated deduction methods were developed for non-classical logics, and resolution was generalised and modified to be applicable to these logics. In general, because of the value of these logics in theoretical computer science and artificial intelligence, a greater awareness of the computational aspects of logical systems was developing and more attention was being devoted to proof-theoretical presentations. It became apparent to us that a key feature in the proof-theoretic study of these logics is that a slight natural variation in an automated or proof-theoretic system of one logic (say $\mathbf{L}_1$), can yield another logic (say $\mathbf{L}_2$).

Although $\mathbf{L}_1$ and $\mathbf{L}_2$ may be conceptually far apart (in their philosophical motivation, and mathematical definitions) when it comes to automated techniques and proof-theoretical presentation, they turn out to be brother and sister. This kind of relationship is not isolated and seems to be widespread. Furthermore, non-monotonic systems seem to be obtainable from monotonic ones through variations on some of their monotonic proof-theoretical formulation, thus giving us a handle on classifying non-monotonic systems.

This phenomena has prompted Gabbay [4, 15] to put forward the view that a logical system $\mathbf{L}$ is not just the traditional consequence relation $\vdash$ (monotonic or non-monotonic) but a pair $(\vdash, \mathbf{S}_\vdash)$, where $\vdash$ is a mathematically defined consequence relation (i.e., the set of pairs (Δ, Q) such that $\Delta \vdash Q$) satisfying whatever minimal conditions on a consequence relation one happens to agree on, and $\mathbf{S}_\vdash$ is an algorithmic system for generating all those pairs. Thus according to this definition classical logic $\vdash$ perceived as a set of tautologies together with a Gentzen system $\mathbf{S}_\vdash$ is not the same as classical logic together with the two-valued truth table decision procedure $\mathbf{T}_\vdash$ for it. In our conceptual framework,$(\vdash, \mathbf{S}_\vdash)$ is *not the same logic* as $(\vdash, \mathbf{T}_\vdash)$.

To illustrate and motivate our way of thinking, observe that it is very easy to move from $\mathbf{T}_\vdash$ for classical logic to a truth table system $\mathbf{T}^n_\vdash$ for Lukasiewicz n-valued logic. It is not so easy to move to an algorithmic system for intuitionistic logic. In comparison, for a Gentzen system presentation, exactly the opposite is true. Intuitionistic and classical logics are neighbours, while Lukasiewicz logics seem completely different. In fact, some of the results of this book show proof-theoretic similarities between Lukasiewicz's infinite valued logic and Girard's Linear Logic, which in turn is proof-theoretically similar to intuitionistic logic.

There are many more such examples among temporal logics, modal logics, defeasible logics and others. Obviously, there is a need for a more unifying framework. The question is then whether we can adopt a concept of a logic where the passage from one system to another is natural, and along predefined acceptable modes of variation? Can we put forward a framework where the computational aspects of a logic also play a role? Is it possible to find a common home for a variety of seemingly different techniques introduced for different purposes in seemingly different intellectual logical traditions?

To find an answer, let us ask ourselves what makes one logic different from another? How is a new logic presented and described and compared to another? The answer is obvious. These considerations are usually dealt with on the meta-level. Most logics are based on modus ponens and the quantifier rules are formally the same anyway and the differences between them are meta-level considerations on the proof theory or semantics. If we can find a mode of presentation of logical systems where meta-level features can reside side by side with object-level features then we can hope for a general framework. We must be careful here. In the logical community the notions of object-level vs. meta-level are not so clear. Most people think of *naming* and *proof predicates* in this connection. This is not what we mean by meta-level here. We need a more refined understanding of the concept. There is a similar need in computer science. In [19] we devote a chapter to these considerations. See also [25].

We found that the best framework to put forward is that of a *Labelled Deductive System, LDS.* Our notion of what constitutes a logic will be that of a pair $(\vdash, \mathbf{S}_\vdash)$ where $\vdash$ is a set-theoretic (possibly non-monotonic) consequence relation on a language $\mathbf{L}$ and $\mathbf{S}_\vdash$ is an *LDS*, and where $\vdash$ is essentially required to satisfy no more than *Identity* (i.e., $\{A\} \vdash A$) and *Surgical Cut* (see below and [20, 21]). This is a refinement of our concept of a logical system mentioned above and first presented in Gabbay [4]. We now not only say that a logical system is a pair $(\vdash, \mathbf{S}_\vdash)$, but we are adding that $\mathbf{S}_\vdash$ itself has a special presentation, that of an *LDS.*

An *LDS* system is a triple $(\mathbf{L}, \Gamma, \mathbf{M})$, where $\mathbf{L}$ is a logical language (connectives and wffs) and Γ is an algebra (with some operations) of labels and $\mathbf{M}$ is a discipline of labelling formulas of the logic (from the algebra of labels Γ), together with deduction rules and with agreed ways of propagating the labels via the application of the deduction rules. The way the rules are used is more or less uniform to all systems. In the general case we allow Γ, the algebra of labels, to be an *LDS* system itself! Furthermore, if our view of a logical system is that the declarative unit is a pair, a formula and a label, then we can also label the pair itself and get multiple labelling.

The perceptive reader may feel resistance to this idea at this stage. First be assured that you are not asked to give up your favourite logic or proof theory nor is there any hint of a claim that your activity is now obsolete. In mathematics a good concept can rarely be seen or studied from one point of view only and it is a sign of strength to have several views connecting different concepts. So the traditional logical views are as valid as ever and add strength to the new point of view. In fact, a closer examination of [19] would reveal that manifestations of our *LDS* approach already exist in the literature in various forms (see [1] and [18] and the references there), however, they were locally regarded as convenient tools and there was not the realisation that there is a general framework to be studied and developed. None of us is working in a vacuum and we build on each others' work. Further, the existence of a general framework in which any particular case can be represented does not necessarily mean that the best way to treat that particular case is within the general framework. Thus if some modal logics can be formulated in *LDS*, this does not mean that in practice we should replace existing ways of treating the logics by their *LDS* formulation. The latter may not be the most efficient for those particular logics. It is sufficient to show how the *LDS* principles specialise and manifest themselves in the given known practical formulation of the logic.

The reader may further have doubts about the use of labels from the computational point of view. What do we mean by a unifying framework? Surely a Turing machine can simulate any logic, is that a unifying framework? The use of labels is powerful, as we know from computer science, are we using labels to play the role of a Turing machine? The answer to the question is twofold. First that we are not operating at the meta-level, but at the object-level. Second, there are severe restrictions on the way we use *LDS*. Here is a preview:

1. The only rules of inference allowed are the traditional ones, modus ponens and some form of deduction theorem for implication, for example.

2. Allowable modes of label propagation are fixed for all logics. They can be adjusted in agreed ways to obtain variations but in general the format is the same. For example, it has the following form for implications:
   $(A \rightarrow B)$ gets label t iff $\forall x \in \Gamma_1$ [If A is labelled x then B can be proved with labels $t + x$], where Γ_1 is a set of labels characterising the implication in that particular logic. For example Γ_1 may be all atomic labels or related labels to t, or variations. The freedom that different logics have is in the choice of Γ_1 and the properties of "+". For example we can restrict the use of modus ponens by a wise propagation of labels.

3. The quantifier rules are the same for all logics.

4. Meta-level features are implemented via the labelling mechanism, which is object language.

The reader who prefers to remain within the traditional point of view of:

assumptions (data) proving a conclusion

can view the labelled formulas as another form of data.

There are many occasions when it is most intuitive to present an item of data in the form $t : A$, where t is a label and A is a formula. The common underlying reason for the use of the label t is that t represents information which is needed to modify A or to supplement (the information in) A which is not of the same type or nature as (the information represented by) A itself. A is a logical formula representing information declaratively, and the additional information of t can certainly be added declaratively to A to form A', however, we may find it convenient to put forward the additional information through the label t as part of a pair $t : A$.

Take for example a source of information which is not reliable. A natural way of representing an item of information from that source is $t : A$, where A is a declarative presentation of the information itself and t is a number representing its reliability. Such expert systems exist (e.g. Mycin) with rules which manipulate both t and A as one unit, propagating the reliability values t_i through applications of modus ponens. We may also use a label naming the source of information and this would give us a qualitative idea of its reliability.

Another area where it is natural to use labels is in reasoning from data and rules. If we want to keep track, for reasons of maintaining consistency and/or integrity constraints, where and how a formula was deduced, we use a label t. In this case, the label t in $t : A$ can be the part of the data which was used to get A. Formally in this case t is a formula, the conjunction of the data used. We thus get pairs of the form $\Delta_i : A_i$, where A_i are formulas and Δ_i are the parts of the database from which A_i was derived.

A third example where it is natural to use labels is time stamping of data. Where data is constantly revised and updated, it is important to time stamp the data items. Thus the data items would look like $t_i : A_i$, where t_i are time stamps. A_i itself may be a temporal formula. Thus there are two times involved, the logical time s_i in $A_i(s_i)$ and the time stamping t_i of A_i. For reasons of clarity, we may wish to regard t_i as a label rather than incorporate it into the logic (by writing for example $A^*(t_i, s_i)$).

To summarise then, we replace the traditional notion of consequence between formulas of the form $A_1, \ldots, A_n \vdash B$ by the notion of consequence between labelled formulas

$$t_1 : A_1, t_2 : A_2, \ldots, t_n : A_n \vdash s : B$$

Depending on the logical system involved, the intuitive meaning of the labels varies. In querying databases, we may be interested in labelling the assumptions so that when we get an answer to a query, we can record, via the label of the answer, from which part of the database the answer was obtained. Another area where labelling is used is temporal logic. We can time stamp assumptions as to when they are true and query, given those assumptions, whether a certain conclusion will be true at a certain time. Thus the consequence notion for labelled deduction is essentially the same as that of any logic: given assumptions does a conclusion follow?

Whereas in the traditional logical system the consequence is defined by using proof rules on the formulas, in the *LDS* methodology the consequence is defined by using rules on both formulas and their labels. Formally we have formal rules for manipulating labels and this allows for more scope in decomposing the various features of the consequence relation. The meta features can be reflected in the algebra or logic of the labels and the object features can be reflected in the rules of the formulas.

The notion of a database or of a "set of assumptions" also has to be changed. A database is a configuration of labelled formulas. The configuration depends on the labelling discipline. For example, it can be a linearly ordered set $\{a_1 : A_1, \ldots, a_n : A_n\}, a_1 < a_2 < \cdots < a_n$. The proof discipline for the logic will specify how the assumptions are to be used. We need to develop the notions of the Cut Rule and the Deduction Theorem in such an environment. This we do in a later section.

The next two sections will give many examples of *LDS* disciplines featuring many known monotonic and non-monotonic logics. It is of value to summarise our view listing the key points involved:

- The unit of declarative data is a labelled formula of the form $t : A$, where A is a wff of a language $\mathbf{L}$ and t is a label. The labels come from an algebra (set) of labels.
- A database is a set of labelled formulas.
- An *LDS* discipline is a system (algorithmic) for manipulating both formulas and their labels. Using this discipline the statement $\Delta \vdash \Gamma$ is well defined for the two databases Δ and Γ. Especially $\Delta \vdash t : A$ is well defined.
- $\vdash$ must satisfy the minimal conditions, namely

  1.3. Identity

  $$\{t : A\} \vdash t : A$$

  1.4. Surgical cut

  $$\frac{\Delta \vdash t : A, \Gamma[t : A] \vdash s : B}{\Gamma[\Delta] \vdash s : B}$$

  where $\Gamma[t : A]$ means that $t : A$ is contained/occurs somewhere in the structure Γ and $\Gamma[\Delta]$ means that Δ replaces A in the structure.

- A logical system is a pair $(\vdash, \mathbf{S}_\vdash)$, where $\vdash$ is a consequence relation and $\mathbf{S}_\vdash$ is an *LDS* for it.

§2. Examples from monotonic logics.

To motivate our approach we study several known examples in this section.

Example 2.1 below shows a standard deduction from Relevance Logic. The purpose of the example is to illustrate our point of view. There are many such examples in Anderson and Belnap [1]. Example 2.3 below considers a derivation in modal logic. There we use labels to denote essentially possible worlds. The objective of the example is to show the formal similarities to the relevance logic case in Example 2.1. Section 2.4 can reap the benefits of the formal similarities of the first two examples and introduce, in the most natural way, a system of relevant modal logic. The objective of Example 2.4 is to show that the labels in Example 2.1 and Example 2.3 can be read as determining the meta-language features of the logic and can therefore be combined "declaratively" to form the new system of Example 2.4. Example 2.5 considers strict implication. This example shows that for strict **S4** implication one can read the labels either as relevance labels or as possible world labels. Examples 2.6, 2.7 show how labels can interact with quantifiers in modal logic.

EXAMPLE 2.1. *Relevance and linear logic.*

Consider a propositional language with implication "$\rightarrow$" only. The forward elimination rule is modus ponens. From the theorem proving view, modus ponens is an object language consideration. Thus a proof of $\vdash (B \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow B))$ can proceed as follows:

Assume $a_1 : B \rightarrow A$ and show $(A \rightarrow B) \rightarrow (A \rightarrow B)$. Further assume $a_2 : A \rightarrow B$ and show $A \rightarrow B$. Further assume $a_3 : A$ and show B. We thus end up with the following problem:

Assumptions

1. $a_1 : B \rightarrow A$
2. $a_2 : A \rightarrow B$
3. $a_3 : A$

Derivation

4. $a_2a_3 : B$ by modus ponens from lines (2) and (3).
5. $a_1a_2a_3 : A$ from (4) and (1).
6. $a_2a_1a_2a_3 : B$ from (5) and (2).
7. $a_2a_1a_2 : A \rightarrow B$ from (3) and (6).
8. $a_2a_1 : (A \rightarrow B) \rightarrow (A \rightarrow B)$ from (2) and (7).
9. $a_2 : (B \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow B))$ from (1) and (8).

The meta aspect of this proof is the annotation of the assumptions and the keeping track of what was used in the deduction. A meta-level condition would determine the logic involved.

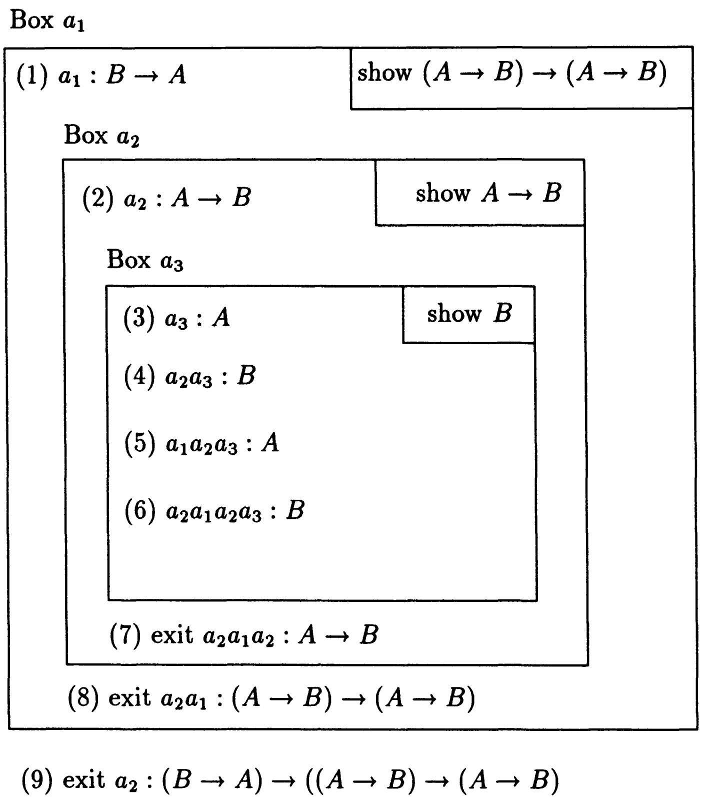
Box a_1
(1) $a_1 : B \to A$
show $(A \to B) \to (A \to B)$
Box a_2
(2) $a_2 : A \to B$
show $A \to B$
Box a_3
(3) $a_3 : A$
show B
(4) $a_2a_3 : B$
(5) $a_1a_2a_3 : A$
(6) $a_2a_1a_2a_3 : B$
(7) exit $a_2a_1a_2 : A \to B$
(8) exit $a_2a_1 : (A \to B) \to (A \to B)$
(9) exit $a_2 : (B \to A) \to ((A \to B) \to (A \to B)$

Figure 1

A formal definition of the labelling discipline for this class of logics is given in [19]. For this example it is sufficient to note the following three conventions:

1. Each assumption is labelled by a new atomic label.

   An ordering on the labels can be imposed, namely $a_1 < a_2 < a_3$. This is to reflect the fact that the assumptions arose from our attempt to prove $(B \to A) \to ((A \to B) \to (A \to B))$ and not for example from $(A \to B) \to ((B \to A) \to (A \to B))$ in which case the ordering would be $a_2 < a_1 < a_3$. The ordering can affect the proofs in certain logics.

2. If in the proof, A is labelled by the multiset α and $A \to B$ is labelled by β then B can be derived with a label $\alpha \cup \beta$ where "$\cup$" denotes multiset union.

3. If B was derived using A as evidenced by the fact that the label α of A is a submultiset of the label β of B ($\alpha \subseteq \beta$) then we can derive $A \to B$ with the label $\beta - \alpha$ ("$-$" is multiset subtraction).

The derivation can be represented in a more graphical way.

To show $(B \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow B))$: See figure 1.

Figure 1 is the *meta-box* way of representing the deduction. Note that in line 8, multiset subtraction was used and only one copy of the label a_2 was taken out. The other copy of a_2 remains and cannot be cancelled. Thus this formula is not a theorem of linear logic, because the outer box does not exit with label $\varnothing$. In relevance logic, the discipline uses sets and not multisets. Thus the label of line 8 in this case would be a_1 and that of line 9 would be $\varnothing$. The above deduction can be made even more explicit as follows:

$(B \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow B))$ follows with a label from Box a_1.

Box a_1

a_1 :	$B \rightarrow A$ assumption
a_2a_1 :	$(A \rightarrow B) \rightarrow (A \rightarrow B)$ from Box a_2

Box a_2

a_2 :	$A \rightarrow B$ assumption
$a_2a_1a_2$:	$A \rightarrow B$ from Box a_3

Box a_3

$a_3 : A$	assumption
$a_2 : A \rightarrow B$	reiteration from box a_2
$a_2a_3 : B$	by modus ponens
$a_1 : B \rightarrow A$	reiteration from box a_1
$a_1a_2a_3 : A$	modus ponens from the two preceding lines
$a_2 : A \rightarrow B$	repetition of an earlier line
$a_2a_1a_2a_3 : B$	modus ponens from the two preceding lines

The following meta-rule was used:

We have a systems of partially ordered meta-boxes $a_1 < a_2 < a_3$. Any assumption in a box a can be reiterated in any box b provided $a < b$.

REMARK 2.2.

a. The above presentation of the boxes makes them look more like possible worlds. The labels are the worlds and formulas can be exported from one world to another according to some rules. The next example 2.3 describes modal logic in just this way.

b. Note that different meta-conditions on labels and meta-boxes correspond to different logics.

The following table gives intuitively some correspondence between meta-conditions and logics.

Meta-condition:	*Logic*
ignore the labels	intuitionistic logic
accept only the derivations which use all the assumptions	relevance logic
accept derivations which use all assumptions exactly once	linear logic

The meta-conditions can be translated into object conditions in terms of axioms and rules. If we consider a Hilbert system with modus ponens and substitution then the additional axioms involved are given below:

Linear Logic
$A \to A$
$(A \to (B \to C)) \to (B \to (A \to C))$
$(C \to A) \to ((B \to C) \to (B \to A))$
$(C \to A) \to ((A \to B) \to (C \to B))$

Relevance Logic
Add the schema below to linear logic
$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$

Intuitionistic Logic
Add the schema below to relevance logic:
$A \to (B \to A)$

The reader can note that the following axiom (Peirce Rule) yields classical logic. Further note that for example, we can define "Linear Classical Logic" by adding the Peirce Rule to linear logic. A new logic is obtained.

Classical Logic
Add the schema below to intuitionistic logic:
$((A \to B) \to A) \to A.$

EXAMPLE 2.3. This example shows the meta-level/object-level division in the case of modal logic. Modal logic has to do with possible worlds. We thus think of our basic database (or assumptions) as a finite set of information about possible worlds. This consists of two parts. The configuration part, the finite configuration of possible worlds for the database, and the assumptions part which tells us what formulas hold in each world. The following is an example of a database:

	Assumptions	*Configuration*
(1)	$t : \Box\Box B$	$t < s$
(2)	$s : \Diamond(B \to C)$	

The conclusion to show (or query) is:

$$t : \Diamond\Diamond C.$$

The derivation is as follows:

3. From (2) create a new point r with $s < r$ and get $r : B \to C$.

We thus have

Assumptions	*Configuration*
(1), (2), (3)	$t < s < r$

4. From (1), since $t < s$ we get $s : \Box B$.

5. From (4) since $s < r$ we get $r : B$.

6. From (5) and (3) we get $r : C$.

7. From (6) since $s < r$ we get $s : \Diamond C$.

8. From (7) using $t < s$ we get $t : \Diamond\Diamond C$.

Discussion:
The object rules involved are:
$\Box E$ *Rule:*

$$\frac{t < s; t : \Box A}{s : A}$$

$\Diamond I$ *Rule:*

$$\frac{t < s, s : B}{t : \Diamond B}$$

$\Diamond E$ *Rule:*

$$\frac{t : \Diamond A}{\text{create a new point } s \text{ with } t < s \text{ and deduce } s : A}$$

Note that the above rules are not complete. We do not have rules for deriving, for example, $\Box A$. Also, the rules are all for intuitionistic modal logic.

The meta-level consideration may be properties of $<$,
e.g. transitivity: $t < s \wedge s < r \rightarrow t < r$ or
e.g. linearity: $t < s \vee t = s \vee s < t$ etc.

EXAMPLE 2.4. The reader can already see the benefit of separating the meta-level (the handling of possible worlds, i.e., labels) and the object-level (i.e., formulas) features. We can combine both the meta-level features of Examples 2.1 and 2.3 to create for example a modal relevance logic in a natural way. Each assumption has a relevance label as well as world label. Thus the proof of the previous example becomes the following:

Assumptions	*Configuration*
(1) $(a_1, t) : \Box\Box B$	$t < s$
(2) $(a_2, s) : \Diamond(B \rightarrow C)$	

We proceed to create a new label r using $\Diamond E$ rule. The relevance label is carried over. We have $t < s < r$.

3. $(a_2, r) : B \rightarrow C$

Using $\Box E$ rule with relevance label carried over, we have:

4. $(a_1, s) : \Box B$

5. $(a_1, r) : B$

Using modus ponens with relevance label updated

6. $(a_1, a_2, r) : C$

Using $\Diamond I$ rule:

7. $(a_1, a_2, s) : \Diamond C$

8. $(a_1, a_2, t) : \Diamond\Diamond C$

(8) means that we got $t : \Diamond\Diamond C$ using both assumptions a_1 and a_2.

There are two serious problems in modal and temporal theorem proving. One is that of Skolem functions for $\exists x \Diamond A(x)$ and $\Diamond \exists x A(x)$ are not logically the same. If we skolemise we get $\Diamond A(c)$. Unfortunately it is not clear where c exists, in the current world $((\exists x = c)\Diamond A(x))$ or the possible world $(\Diamond(\exists x = c)A(x))$.

If we use labelled assumptions then, $t : \exists x \Diamond A(x)$ becomes $t : \Diamond A(c)$ and it is clear that c is introduced at t.

On the other hand, the assumption $t : \Diamond \exists x A(x)$ will be used by the $\Diamond E$ rule to introduce a new point $s, t < s$ and conclude $s : \exists x A(x)$. We can further skolemise at s and get $s : A(c)$, with c introduced at s. We thus need the mechanism of remembering or labelling constants as well, to indicate where they were first introduced.

Labelling systems for modal and temporal logics is studied in [22].

EXAMPLE 2.5. The following example describes the logic of modal **S4** strict implication. In this logic the labels can be read either as relevance labels or as possible worlds. **S4** strict implication $A \rightarrow B$ can be understood as a temporal connective, as follows:

"$A \rightarrow B$ is true at world t iff for all future worlds s to t and for t itself we have that if A is true at s then B is true at s". Thus $A \rightarrow B$ reads "From now on, if A then B".

Suppose we want to prove that $A \rightarrow B$ and $A \rightarrow (B \rightarrow C)$ imply $A \rightarrow C$. To show this we reason semantically and assume that at time t, the two assumptions are true. We want to show that $A \rightarrow C$ is also true at t. To prove that we take any future time s, assume that A is true at s and show that C is also true at s. We thus have the following situation:

1. $t : A \rightarrow B$

2. $t : A \rightarrow (B \rightarrow C)$

3. show $t : A \rightarrow C$
   from box

3.1	Assume $s : A$ Show $s : C$ Since s is in the future of t, we get that at s, (1) and (2) are also true.
3.2	$s : A \to B$ from (1)
3.3	$s : A \to (B \to C)$ from (2) We now use modus ponens, because $X \to Y$ means "from now on, if X then Y"
3.4	$s : B$ from (3.1) and (3.2)
3.5	$s : B \to C$ from (3.2) and (3.3)
3.6	$s : C$ modus ponens from (3.4) and (3.5)

exit $t : A \to C$

Notice that any $t : D$ can be brought into (reiterated) the box as $s : D$, provided it has an implicational form, $D = D_1 \to D_2$. We can thus regard the labels above as simply naming assumptions (not as possible worlds) and the logic has the reiteration rule which says that only implications can be reiterated.

Let us add a further note to sharpen our understanding. Suppose $\to$ is read as a **K4** implication (i.e., transitivity without reflexivity). Then the above proof should fail. Indeed the corresponding restriction on modus ponens is that we do perform $X, X \to Y \vdash Y$ in a box, provided $X \to Y$ is a reiteration into the box and was not itself derived in that same box. This will block line (3.6).

EXAMPLE 2.6. Another example has to do with the Barcan formula

Assumption	*Configuration*
(1) $t : \forall x \Box A(x)$	$t < s$

We show
$$s : \forall x A(x)$$
We proceed intuitively

1. $t : \Box A(x)$ (stripping $\forall x$, remembering x is arbitrary).
2. Since the configuration contains $s, t < s$ we get
$$s : A(x)$$
3. Since x is arbitrary we get
$$s : \forall x A(x)$$

The above intuitive proof can be restricted.
The rule
$$\frac{t : \Box A(x), t < s}{s : A(x)}$$
is allowed only if x is instantiated.

To allow the above rule for arbitrary x is equivalent to adopting the Barcan formula axiom:
$$\forall x \Box A(x) \to \Box \forall x A(x)$$

EXAMPLE 2.7. To show $\forall x \Box A(x) \to \Box \forall x A(x)$ in the modal logic where it is supposed to be true.

1. Assume $t : \forall x \Box A(x)$
   We show $\Box\forall x A(x)$ by the use of the meta-box:

	create α,	$t < \alpha$
(2)	$t : \Box A(x)$	from (1)
(3)	$\alpha : A(x)$	from (2) using a rule which allows this with x a variable.
(4)	$\alpha : \forall x A(x)$	universal generalisation.

(5) Exit: $t : \Box\forall x A(x)$.

This rule has the form:

Create α,	$t < \alpha$
Argue to get	$\alpha : B$
Exit with	$t : \Box B$

The above are just a few examples for the scope we get using labels. The exact details and correspondences are worked out in our monograph [19].

EXAMPLE 2.8. (Relevance reasoning.) The indices are α, β, and $\gamma = (\beta - \alpha)$. The reasoning structure is:
Assume $\alpha : A$
Show $\beta : B$
If $\beta \supseteq \alpha$ then exit with $(\beta - \alpha) : A \to B$.
To show $A \to (B \to C) \vdash B \to (A \to C)$

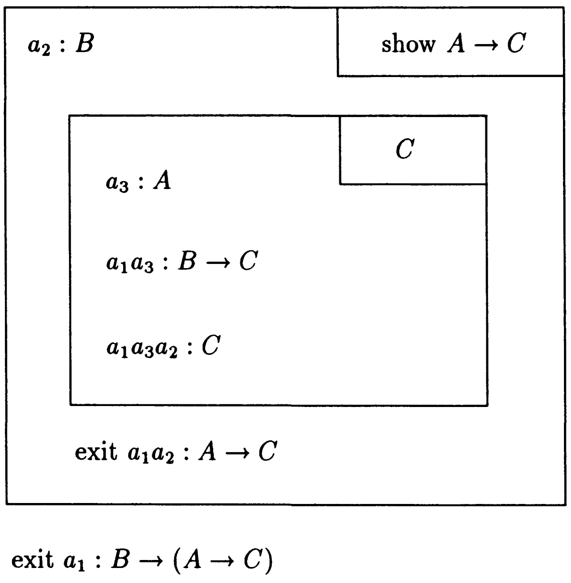

Figure 2

Assume

$$a_1 : A \to (B \to C)$$

we use the meta-box to show $B \to (A \to C)$. See figure 2.

EXAMPLE 2.9. (Lukasiewicz many-valued logics.) Consider Lukasiewicz infinite-valued logic, where the values are all real numbers or rationals in [0,1]. We designate 0 as *truth* and the truth table for implication is

$$x \to y = \max(0, y - x)$$

Here the language contains atoms and implication only, assignments h give values to atoms in [0,1], $h(q) \in [0,1]$ and h is extended to arbitrary formulas via the table for $\to$ above. Define the relation

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

to mean that for all $h, h(A_1) + \cdots + h(A_n) \geqq h(B)$, where $+$ is numerical addition.

This logic can be regarded as a labelled deductive system, where the labels are values $t \in [0,1]$. $t : A$ means that $h(A) = t$, for a given background assignment h. The interesting part is that to show $t : A \to B$ (i.e., that $A \to B$ has value t) we assume $x : A$ (i.e., that A has value x) and then have to show that B has value $t + x$, i.e., show $t + x : B$.

This is according to the table of $\to$.

Thus figure 3 shows the deduction in box form:

$$\boxed{\begin{array}{ll} x : A & \text{assumption} \\ \vdots & \\ \vdots & \\ t + x : B & \end{array}}$$

exit $t : A \to B$

Figure 3

This has the *same structure* as the case of relevance logic, where $+$ was understood as concatenation.

A full study of many valued logics from the LDS point of view is given in [19].

EXAMPLE 2.10. (Formulas as types.) Another instance of the natural use of labels is the Curry–Howard interpretation of formulas as types. This interpretation conforms exactly to our framework. In fact, our framework gives the incentive to extend the formulas as types interpretation in a natural way to other logics, such as linear and relevance logics and surprisingly, also many valued logics, modal logics, and intermediate logics. A formula is considered as a type and its label is a *definable* λ-term of the same type. Given a system for defining λ-terms, the theorems of the logic are all those types which can be shown to be non-empty.

The basic propagation mechanism corresponding to modus ponens is:

$$\frac{\begin{array}{l} t^A : A \\ t^{A\to B} : A \to B \end{array}}{t^{A\to B}(t^A) : B}$$

It is satisfied by ***application.***

Thus if we read the $+$ in $t^{A\to B} + t^A$ as application, we get the exact parallel to the general schema of propagation. Compare with relevance logic where $+$ was concatenation, and with many valued logics where $+$ was numerical addition!

To show $t : A \to B$ we assume $x : A$, with x arbitrary, i.e., start with a term x of type A, use the proof rules to get B. As we saw, applications of modus ponens generate more terms which contain x in them via application. If we accept that proofs generate functionals, then we get B with a label $y = t(x)$. Thus $t = \lambda x t(x)$. This again conforms with our general schema for $\to$.

In our paper [18] on the Curry–Howard interpretation we exploit this idea systematically. There are two mechanisms which allow us to restrict or expand our ability to define terms of any type. We can restrict λ-abstraction, (e.g. allow $\lambda x t(x)$ only if x actually occurs in t), this will give us logics weaker than intuitionistic logic, or we can increase our world of terms by requiring diagrams to be closed e.g., for any φ of classical logic such that

$$\vdash (A \to B) \to [\varphi(A) \to \varphi(B)]$$

in classical logic, we want the following diagram to be complete, i.e., for any term t there must exist a term t' (see figure 4).

$$\begin{tikzcd}
\varphi(A) \arrow[r, "t'"] & \varphi(B) \\
A \arrow[u] \arrow[r, "t"] & B \arrow[u]
\end{tikzcd}$$

Figure 4

Take for example the formula $A \to (B \to A)$ as type. We want to show a definable term of this type, we can try and use the standard proof (see figure 5),

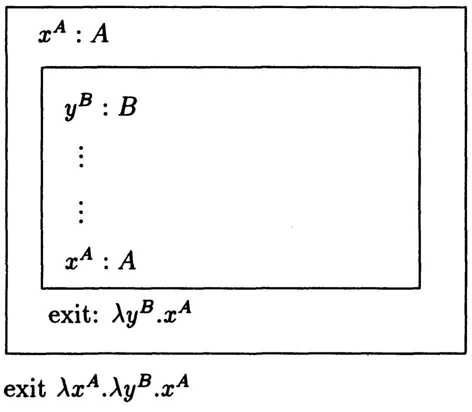

Figure 5

however, with the restriction on λ-abstraction which requires the abstracted variable to actually occur in the formula, we cannot exit the inner box. For details see [18].

EXAMPLE 2.11. (Realisability interpretation.) The well-known realisability interpretation for intuitionistic implication is another example of a functional interpretation for $\to$ which has the same universal *LDS* form. A notation for a recursive function $\{e\}$ realises an implication $A \to B$ iff for any n which realises $A, \{e\}(n)$ realises B. Thus

$$e : A \to B \text{ iff } \forall n[n : A \Rightarrow \{e\}(n) : B]$$

It is an open problem to find an axiomatic description of the set of all wffs which are realisable.

§3. Examples from non-monotonic logics.

The examples in the previous section are from the area of monotonic reasoning. This section will give examples from non-monotonic reasoning. As we have already mentioned, we hope that the idea of *LDS* will unify these two areas.

EXAMPLE 3.1. (Ordered logic.) An ordered logic database is a partially ordered set of local databases, each local database being a set of clauses. The diagram (figure 6) describes an ordered logic database.

The local databases are labelled t_1, t_2, t_3, s_1, s_2 and $\varnothing$ and are partially ordered as in the figure.

To motivate such databases, consider an ordinary logic program $C_1 = \{p \leftarrow \neg q\}$. The computation of a logic program assumes that, since q is not a head of any clause, $\neg q$ is part of the data (this is the *closed world assumption*). Suppose we relinquish this principle and adopt the principle of asking an *advisor* what to do with $\neg q$. The advisor might say that $\neg q$ succeeds or might say that $\neg q$ fails. The advisor might have his own program to consult. If his program is C_2, he might run the goal q (or $\neg q$), look at what he gets and then advise. To make the situation symmetrical and general we must allow for Horn programs to have rules with both q and $\neg q$ (i.e., literals) in heads and bodies and have any number of negotiating advisors. Thus we can have $C_2 = \{\neg q\}, C_1 = \{q \leftarrow \neg q\}$ and C_1 depends on C_2. Ordered logic develops and studies various aspects of such an advisor system which is modelled as a partially ordered set of theories. Such a logic is useful, e.g. for multi-expert systems where we want to represent the knowledge of several experts in a single system. Experts may then be ordered according to an "advisory" or a relative preference relation.

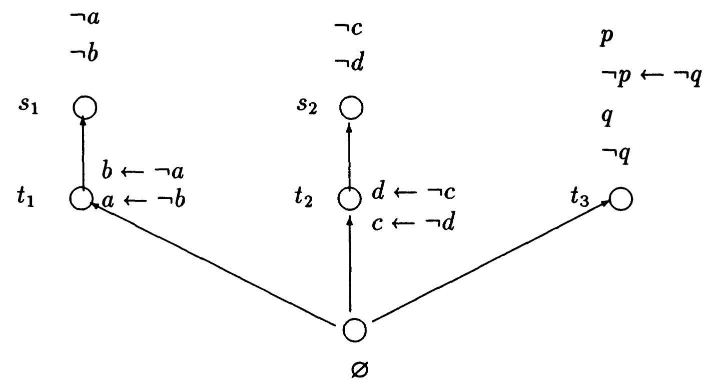

Figure 6

A problem to consider is what happens when we have several advisors that are in conflict. For example, C_1 depends on C_2 and C_1 depends on C_3. The two advisers, C_2 and C_3, may be in conflict. One may advise $\neg q$, the other q. How to decide? There are several options:

1. We can accept q if all advisors say "yes" to q.

2. We can accept q if at least one advisor says "yes" to q.

3. We can apply some non-monotonic or probabilistic mechanism to decide.

If we choose options (1) or (2) we are essentially in modal logic. To have a node t and to have $?q$ refer to advisors $t_1, \ldots, t_n$ with $t < t_i, i = 1, \ldots, n$ is like considering $?\Box q$ at t in modal logic with $t_1, \ldots, t_n$ possible worlds in option 1 and like considering $\Diamond q$ at t in option (2). Option (3) is more general, and here an *LDS* approach is most useful. We see from this advisors examples an application area where the labels arise naturally and usefully. The area of ordered logic is surveyed in [13].

EXAMPLE 3.2. (Defeasible logic.) This important approach to non-monotonic reasoning was introduced by Nute [14]. The idea is that rules can prove either an atom q or its negation $\neg q$. If two rules are in conflict, one proving q and one proving $\neg q$, the deduction that is stronger is from a rule whose antecedent is logically more specific. Thus the database:

$$\begin{aligned} t_1 &: \text{Bird}\ (x) \rightarrow \text{Fly}\ (x) \\ t_2 &: \text{Big}\ (x) \wedge \text{Bird}\ (x) \rightarrow \neg\ \text{Fly}\ (x) \\ t_3 &: \text{Big}\ (a) \\ t_4 &: \text{Bird}\ (a) \end{aligned}$$

$$\begin{array}{l} t_1 < t_2 \\ \quad t_3 \\ \quad t_4 \end{array}$$

can prove:

$$\begin{aligned} t_2t_3t_4 &: \neg\text{Fly}(a) \\ t_1t_4 &: \text{Fly}(a) \end{aligned}$$

The database will entail $\neg$Fly (a) because the second rule is more specific.

As an *LDS* system the labelling of rules in a database Δ is very simple. We label a rule by its antecedent. The ordering of the labels is done by logical strength relative to some background theory Θ (which can be a subtheory of Δ of some form). Deduction pays attention to strength of labels.

EXAMPLE 3.3. (Propositional circumscription.) Circumscription is defined semantically via satisfaction in minimal models. Surprisingly, results of Olivetti [26] allow one to present an *LDS* discipline for (at least) propositional circumscription.

To explain the idea let $\vdash_m$ denote consequence in minimal models. For this consequence we have, for example, $p \vee q \vdash_m \neg p \vee \neg q$, which does not follow in classical logic. Suppose we try and find a semantic tableaux counter-model for the above. In classical logic we try the tableaux construction and if all the top nodes are *closed* then there is no countermodel. For $\vdash_m$ we just change the notion of "*closed*." This can depend on labelling. A more precise study of this theme will be done later.

§4. Conclusion.

Logic is widely applied in computer science and artificial intelligence. The needs of the application areas in computing are different from those in mathematics and philosophy. In response to computer science needs, intensive research has been directed in the area of non-classical and non-monotonic logic. New logics have been developed and studied. Certain logical features, which have not received extensive attention in the pure logic community, are repeatedly being called upon in computational applications. Two features in logic seem to be of crucial importance to the needs of computer science and stand in need of further study. These are:

1. The meta-level features of logical systems

2. The "logic" of Skolem functions and unification

The meta-language properties of logical systems are usually hidden in the object language. Either in the proof theory or via some higher-order or many-sorted devices. The logic of Skolem functions is non-existent. Furthermore, the traditional presentation of classical and non-classical logics is not conducive to bringing out and developing the features needed for computer science applications. The very concept of what is a logical system seems to be in need of revision and clarification. A closer examination of classical and non-classical logics reveals the possibility of introducing a new approach to logic; the discipline of *Labelled Deductive Systems* (*LDS*) which, I believe, will not only be ideal for computer science applications but will also serve, I hope, as a new unifying logical framework of value to logic itself. What seem to be isolated local features of some known logics turn out to be, in my view, manifestations of more general logical phenomena of interest to the future development of logic itself.

LDS is part of a more general view of logic. This view is discussed elsewhere [19, 20, 21], however in brief, we claim the following. The new concept of a logical system is that of a *network* of *LDS* systems which has mechanisms for *communication* (through the labels, which code meta-information) and *evolution* or change.

Evaluation is a general concept which can embrace updating, abduction, consistency maintenance, action and planning. The above statement of position is vague but it does imply that we believe that notions like abduction and updating are logical notions of equal standing to those of provability. See [17].

References

[1] A. R. Anderson and N. D. Belnap, ***Entailment***, Princeton University Press, 1975.

[2] D. M. Gabbay, ***Semantical Investigations in Heyting's Intuitionistic Logic***, D. Reidel, 1981.

[3] D. M. Gabbay, *Theoretical Foundations for non monotonic reasoning*, in ***Expert Systems, Logics and Models of Concurrent Systems***, K. Apt (ed.), Springer-Verlag, 1985, pp. 439–459.

[4] D. M. Gabbay, *Theory of Algorithmic Proof*, in ***Handbook of Logic in Computer Science***, Volume 1, S. Abramsky, D. M. Gabbay, T. S. E. Maibaum (eds.), Oxford University Press, 1993, pp. 307–408.

[5] D. Makinson, *General Theory of Cumulative Inference*, in ***Non-monotonic Reasoning***, M. Reinfrank, J. de Kleer, M. L. Ginsberg and E. Sandewall (eds.), Lecture Notes on Artificial Intelligence, no. 346, Springer-Verlag.

[6] D. Makinson, *General Patterns in nonmonotonic Reasoning*, in ***Handbook of Logic in Artificial Intelligence and Logic Programming***, Volume 2, D. M. Gabbay, C. J. Hogger, J. A. Robinson (eds.), Oxford University Press, to appear, 1993.

[7] D. SCOTT, *Completeness and axiomatizability in many valued logics*, in ***Proceedings of the Tarski Symposium***, American Mathematical Society, 1974, pp. 411–436.

[8] A. TARSKI, *On the Concept of Logical Consequence* (in Polish), 1936. Translation in ***Logic Semantics Metamathematics***, Oxford University Press, 1956.

[9] R. WOJCICKI, *An Axiomatic treatment of non monotonic arguments*, ***Studia Logica***, to appear.

[10] R. WOJCICKI, *Heuristic Rules of Inference in non-monotonic arguments*, ***Studia Logica***, to appear.

[11] S. KRAUS, D. LEHMANN, and M. MAGIDOR, *Preferential models and cumulative logics*, ***Artificial Intelligence***, vol. 44 (1990), pp. 167–207.

[12] D. LEHMANN, *What does a conditional knowledge base entail?* in ***KR 89, Toronto, May 89***, Morgan Kaufmann Publisher, pp. 1–18.

[13] D. VERMEIR and E. LAENENS, *An overview of ordered logic*, in ***Abstracts of the Third Logical Biennial***, Varga, Bulgaria, 1990.

[14] D. NUTE, *LDR—A Logic for Defeasible Reasoning*, 1986, ACMC Research Report 01-0013.

[15] D. M. GABBAY, *Algorithmic Proof with Diminishing Resource, I*, in ***Proceedings CSL 90***, LNCS 533, Springer-Verlag, pp. 156–173,

[16] D. M. GABBAY, *The Craig Interpolation Theorem for Intuitionisic Logic I* and *II*, in ***Logic Colloquium 69***, R. O. Gandy (ed.), North-Holland Pub. Company, pp. 391–410.

[17] D. M. GABBAY, *Abduction in labelled deductive systems, a conceptual abstract*, in ***ECSQAU 91***, R. Kruse and P. Siegel (eds.), Lecture notes in Computer Science 548, Springer-Verlag, 1991, pp. 3–12.

[18] D. M. GABBAY and R. J. G. B. DE QUEIROZ, *Extending the Curry–Howard Interpretation to Linear, Relevant and other Resource Logics*, ***The Journal of Symbolic Logic***, vol. 57 (1992), pp. 1319–1365.

[19] D. M. GABBAY, *Labelled Deductive Systems*, 1st Draft September 1989, 6th draft February 1991. Published as a report by CIS, University of Münich. To appear as a book with Oxford University Press.

[20] D. M. GABBAY, *Theoretical Foundations for non monotonic reasoning Part 2: Structured non-monotonic Theories*, in ***SCAI '91***, Proceedings of the Third Scandinavian Conference on AI, IOS Press, Amsterdam, pp. 19–40.

[21] D. M. GABBAY, *A General Theory of Structured Consequence Relations*, to appear in a volume on substructured logics, P. Schröder-Heister and K. Dosen (eds.), Oxford University Press.

[22] D. M. GABBAY, *Modal and Temporal Logic Programming II*, in ***Logic Programming—Expanding the Horizon***, T. Dodd, R. P. Owens, S. Torrance (eds.), Ablex, 1991, pp. 82–123.

[23] D. M. GABBAY, *How to construct a logic for your application*, in ***Proceedings of the 16th German AI Conference, GWAI 92***, Lecture Notes on AI, vol. 671, Springer-Verlag, 1992, pp. 1–30.

[24] D. M. GABBAY, *Labelled Deductive Systems and Situation Theory*, to appear in ***Proceedings STA-III***, 1992.

[25] D. M. GABBAY, *Modal and Temporal Logic Programming III, Metalevel Features in the Object Language*, in ***Non-Classical Logic Programming***, L. Fariñas del Cerro and M. Penttonen (eds.), Oxford University Press, 1992, pp. 85–124.

[26] N. OLIVETTI, *Tableaux and Sequent Calculus for Minimal Entailment*, ***Journal of Automated Reasoning***, vol. 9 (1992), pp. 99–139.

Department of Computing
Imperial College, 180 Queen's Gate
London SW7 2BZ.
e-mail: dg@doc.ic.ac.uk
Tel: 071 589 5111