Generating Explicit Orderings for Non-monotonic Logics

James Cussens
Centre for Logic and Probability in IT
King's College
Strand
London, WC2R 2LS, UK
j.cussens@elm.cc.kcl.ac.uk

Anthony Hunter
Department of Computing
Imperial College
180, Queen's Gate
London, SW7 2BZ, UK
abh@doc.ic.ac.uk

Ashwin Srinivasan
Programming Research Group
Oxford University Computing Laboratory
11, Keble Road
Oxford, OX1 3QD, UK
ashwin.srinivasan@prg.ox.ac.uk

Abstract

For non-monotonic reasoning, explicit orderings over formulae offer an important solution to problems such as 'multiple extensions'. However, a criticism of such a solution is that it is not clear, in general, from where the orderings should be obtained. Here we show how orderings can be derived from statistical information about the domain which the formulae cover. For this we provide an overview of prioritized logics—a general class of logics that incorporate explicit orderings over formulae. This class of logics has been shown elsewhere to capture a wide variety of proof-theoretic approaches to non-monotonic reasoning, and in particular, to highlight the role of preferences—both implicit and explicit—in such proof theory. We take one particular prioritized logic, called SF logic, and describe an experimental approach for comparing this logic with an important example of a logic that does not use explicit orderings of preference—namely Horn clause logic with negation-as-failure. Finally, we present the results of this comparison, showing how SF logic is more skeptical and more accurate than negation-as-failure.

Keywords: non-monotonic reasoning, statistical inference, prioritized logics, machine learning.

Introduction

Within the class of non-monotonic logics and associated systems such as inheritance hierarchies, there is a dichotomy between those formalisms that incorporate explicit notions of preference over formulae, and those that do not. Even though using explicit orderings offers an effective mechanism for obviating certain kinds of 'multiple extension' problems, their use remains controversial. A major criticism is that it is unclear where the orderings come from. We address this criticism by arguing that the orderings should be derived from statistical information generated from the domain over which they operate. If we delineate the kind of information about the domain that we require, then there are generic mappings from this information into the set of orderings over data.

The structure of the paper is as follows. First, we provide an overview of prioritized logics—a general class of logics that incorporate explicit preferences over formulae. Second, we take one particular prioritized logic, SF logic, and describe an experimental approach to comparing this logic with an important example of a logic that does not use explicit orderings of preference—namely Horn clause logic with negation-as-failure. Third, we take SF logic and show how we can generate explicit orderings over the data using statistical inference, and finally, we present the results of the comparison, showing how SF logic is more skeptical and more accurate than negation-as-failure.

Overview of prioritized logics

In order to provide a general framework for logics with explicit orderings, we use the family of prioritized logics [Hunter, 1992; Hunter, 1993]. Within this family, each member is such that:

- Each formula of the logic is labelled.
- The rules of inference for the logic are augmented with rules of manipulation for the labels.
- The labels correspond to a partially-ordered structure.

A prioritized logic is thus an instance of a labelled deductive system [Gabbay, 1991a; Gabbay, 1991b; Gabbay and de Queiroz, 1993].

A prioritized logic can be used for non-monotonic reasoning by defining a consequence relation that allows the inference of the formula with a label that is 'most preferred' according to some preference criterion. Furthermore, we can present a wide variety of existing non-monotonic logics in this framework. In particular, we can explore the notion of implicit or explicit preference prevalent in a diverse class of non-monotonic logics. For example, by adopting appropriate one-to-one rewrites of formulae into prioritized logic formulae, we can show how the propositional form of a number of key non-monotonic logics including negation-as-failure with general logic programs, ordered logic, LDR, and a skeptical version of inheritance hierarchies, can be viewed as using preferences in an essentially equivalent fashion [Hunter, 1993].

For this paper, we use a member of the prioritized logics family called SF logic and defined as follows. The language is composed of labelled formulae of the following form, where $\alpha_0, \ldots, \alpha_n, \beta$ are unground literals, i is a unique label, and $n \geq 0$.

$$i : \alpha_0 \wedge \ldots \wedge \alpha_n \rightarrow \beta$$

We call these formulae SF *rules*. We also allow uniquely labelled ground positive and negative literals which we call SF *facts*. A database Δ is a tuple $(\Gamma, \Omega, \succeq, \sigma)$, where Γ is a set of SF rules and facts, Ω is $[0,1] \times [0,1]$, $\succeq$ is some partial-ordering relation over Ω, and σ is a map from labels into Ω. This means each label corresponds to a pair of reals, and in this sense the ordering is two-dimensional. We use $[0,1] \times [0,1]$ only as a convenient representation for two-dimensional partial orderings. For the SF facts, σ maps to $(1,1)$.

There is more than one intuitive way of combining these two dimensions of values to generate a single poset $(\Omega, \succeq)$. We define the $\succeq$ relation in terms of $\geq$, the usual ordering relation for the real numbers. Consider the following definitions for the $\succeq$ relation.

Definition 1 $(i,p) \succ (j,q)$ *iff* $(i > j)$ *or* $(i = j$ *and* $p > q)$

Definition 2 $(i,p) \succ (j,q)$ *iff* $(i > j$ *and* $p > q)$

Definition 1 imposes a total ordering on Ω, where the ordering on the first label takes precedence, and the second label is only used as a 'tie-breaker'. Definition 2 defines a non-total subset of the first relation, where both dimensions play an equally important rôle.

The $\succ$ relation can be used to resolve conflicts in the arguments that emanate from Γ and is used as such in the consequence relation for a prioritized logic. The SF consequence relation $\mid\sim$ allows an inference α if α is *proposed* and *undefeated* or if α is a fact. It is *proposed* if and only if there is an argument for α, such that all conditions for α are satisfied by recursion. It is *undefeated* if and only if there is no more preferred arguments for the complement of α. For all databases Δ, atomic labels i, unground literals α, groundings μ, and ground literals δ, the $\mid\sim$ relation is defined as follows, where, if δ is a positive literal then $\hat{\delta} = \neg\delta$ and $\widehat{\neg\delta} = \delta$:

Definition 3

$\Delta \mid\sim \delta$ *if* $i : \delta \in \Delta$ *and* $\sigma(i) = (1,1)$

$\Delta \mid\sim \delta$ *if* $\exists i[proposed(\Delta, i, \delta)$ *and* $undefeated(\Delta, i, \delta)]$

$proposed(\Delta, i, \delta)$ *iff*
$\exists \beta_0, \ldots, \beta_n, \mu$ $[i : \beta_0 \wedge \ldots \wedge \beta_n \rightarrow \alpha \in \Delta$
and $\mu(\alpha) = \delta$
and $\Delta \mid\sim \mu(\beta_0), \ldots, \Delta \mid\sim \mu(\beta_n)]$

$undefeated(\Delta, i, \delta)$ *iff*
$\forall j[proposed(\Delta, j, \hat{\delta}) \Rightarrow \sigma(i) \succ \sigma(j)]$

Suppose $\Gamma = \{r : \alpha(a),\ p : \alpha(x) \rightarrow \beta(x),\ q : \alpha(x) \rightarrow \neg\beta(x)\}$, where $\sigma(p) = (0.6, 0.7)$, $\sigma(q) = (0.5, 0.8)$ and $\sigma(r) = (1,1)$. Using Definition 1, $\sigma(p) \succ \sigma(q)$, so $\Delta \mid\sim \beta(a)$ and $\Delta \mid\sim \alpha(a)$ hold. Using Definition 2, however, we have that $\sigma(p) \not\succ \sigma(q)$ and $\sigma(q) \not\succ \sigma(p)$ so $\Delta \not\mid\sim \beta(a)$ and $\Delta \not\mid\sim \neg\beta(a)$, but still $\Delta \mid\sim \alpha(a)$. This illustrates how Definition 2 captures a more skeptical logic.

Generating Rules

To show the value of generating explicit orderings, we now want to compare SF logic with existing non-monotonic logics. However, instead of undertaking this comparison purely on theoretical grounds, we do an empirical comparison with Horn clause logic augmented with negation-as-failure (Prolog). To support this, we use a machine learning algorithm, Golem [Muggleton and Feng, 1990], to generate definite clauses. Golem is an inductive logic programming approach to learning [Muggleton, 1991; Muggleton, 1992]. Using Golem means that significantly large numbers of examples can be used to generate these clauses. This facilitates the empirical study, and supports the statistical inference used for generating the explicit ordering. The definite clauses are used directly by Prolog, and used via a rewrite by SF logic.

We assume a set of ground literals D which express relevant facts about the domain in question and we also assume a target predicate symbol β. Since Prolog does not allow classical negation, we adopt the following non-logical convention: for a predicate symbol β, we represent negative examples using the predicate symbol not_β. Golem learns definite clauses where the head of the clause has the target predicate symbol β. This is done by using a training set Tr of N randomly chosen

literals from D, where each of these literals has either β or not_β as predicate symbol. To learn these clauses Golem uses background knowledge, which is another subset of D, where none of the literals has β or not_β as a predicate symbol. The literals in the antecedents of the learnt clauses use the predicate symbols from the background knowledge.

For example, given background knowledge $\{\alpha(c_1), \alpha(c_2), \alpha(c_4), \alpha(c_7), \delta(c_1)\}$ and training examples $\{\beta(c_1), \beta(c_2), \text{not\_}\beta(c_3)\}$, Golem would learn the clause $\alpha(x) \rightarrow \beta(x)$, which has *training accuracy* 100%. In practice we use significantly larger sets of background knowledge and training examples. Also, we allow Golem to induce clauses with training accuracy below 100%, since learning completely accurate rules is unrealistic in many domains.

The induced clauses are tested using testing examples—ground literals with predicate symbol either β or not_β, which are randomly selected from $D \setminus Tr$. In the normal execution of Golem these testing examples are treated as queries to the induced set of definite clauses and are evaluated using $\vdash_p$, the Prolog consequence relation.

We define a function f to evaluate each test example. f takes Δ, the union of the learnt rules and the background knowledge, and the test example $\beta(\bar{c})$ or not_$\beta(\bar{c})$, where $\bar{c}$ is a tuple of ground terms and returns an evaluation of Δ's prediction concerning the test example. (Recall that none of the ground literals in the background knowledge have β as a predicate symbol.)

Definition 4

$$\begin{array}{lcll} f(\Delta, \beta(\bar{c})) & = & correct & \text{if } \Delta \vdash_p \beta(\bar{c}) \\ f(\Delta, \beta(\bar{c})) & = & incorrect & \text{if } \Delta \nvdash_p \beta(\bar{c}) \\ f(\Delta, not\_\beta(\bar{c})) & = & correct & \text{if } \Delta \nvdash_p \beta(\bar{c}) \\ f(\Delta, not\_\beta(\bar{c})) & = & incorrect & \text{if } \Delta \vdash_p \beta(\bar{c}) \end{array}$$

To continue the above example, suppose that $\{\beta(c_4), \text{not\_}\beta(c_5), \beta(c_6), \text{not\_}\beta(c_7)\}$ were the test examples, then we would have $f(\Delta, \beta(c_4)) = f(\Delta, \text{not\_}\beta(c_5))$ = correct and $f(\Delta, \beta(c_6)) = f(\Delta, \text{not\_}\beta(c_7))$ = incorrect. The clause $\alpha(x) \rightarrow \beta(x)$ would then have the extremely poor *test accuracy* of 50%.

We generate SF rules in two stages. First, we run Golem with target predicate β, then we rerun it with target predicate not_β. To get the SF rules, we take the union of the two sets of clauses, rewrite the not_β symbol to the negated symbol $\neg\beta$, uniquely label each clause, and provide a map σ from the labels into Ω. This map is determined by information contained in the training data, and methods for defining it are described in the next section.

Let Δ denote the union of the SF rules with the background data. The examples from the test set are then used to query Δ. Suppose $\gamma(\bar{c})$ were such an example, where either $\gamma = \beta$ or $\gamma = \neg\beta$, then one of the following obtains (1) $\Delta \mathrel{|\!\sim} \gamma(\bar{c})$; (2) $\Delta \mathrel{|\!\sim} \widehat{\gamma(\bar{c})}$; or (3) ($\Delta \not\mathrel{|\!\sim} \gamma(\bar{c})$ and $\Delta \not\mathrel{|\!\sim} \widehat{\gamma(\bar{c})}$). We define the function g to evaluate each example (note that for SF logic we have extra category 'undecided').

Definition 5

$$\begin{array}{lcll} g(\Delta, \gamma(\bar{c})) & = & correct & \text{if } \Delta \mathrel{|\!\sim} \gamma(\bar{c}) \\ g(\Delta, \gamma(\bar{c})) & = & incorrect & \text{if } \Delta \mathrel{|\!\sim} \widehat{\gamma(\bar{c})} \\ g(\Delta, \gamma(\bar{c})) & = & undecided & \text{if } \Delta \not\mathrel{|\!\sim} \gamma(\bar{c}), \Delta \not\mathrel{|\!\sim} \widehat{\gamma(\bar{c})} \end{array}$$

Generating preference orderings

We now describe how to elicit a preference ordering over formulae by using facts about the domain, specifically facts from that subset of them which constitutes the training set of examples. To construct a preference ordering over the induced formulae, we find a pair of values which measure how well *confirmed* each SF rule $i : \alpha \rightarrow \beta$ is by the training data. We then map the unique label associated with each SF rule to this pair of values in Ω via the mapping σ.

For the first value, we calculate an estimate, denoted $\tilde{p}$, of the probability $P(\beta|\alpha) = p$. p is the probability that a (randomly chosen) example, given that it satisfies α, also satisfies β. Equivalently, it is the proportion of those examples that satisfy α which also satisfy β. p is an obvious choice as a measure of preference, it is the probability that the rule $\alpha \rightarrow \beta$ correctly classifies examples which it covers, i.e. examples which satisfy its antecedent α. Unfortunately, the value p can not be determined without examining all individuals in the domain that satisfy α and determining what proportion of them also satisfy β. This is infeasible for any domain large enough to be of interest. We show below how various estimates $\tilde{p}$ are constructed.

Clearly, we want $\tilde{p}$ close to p, i.e. we want to minimise $l = |\tilde{p} - p|$. The value l would be an ideal measure of the precision of the estimate $\tilde{p}$, but it will be unknown, since p is unknown. Instead, we either use *relative cover* (see below) or $P(l < t)$ for some fixed t, as a measure of the reliability of $\tilde{p}$. This gives us our second value.

For details of the various estimates used, see [Cussens, 1993]. We give only a brief sketch here, since the important point is that we can use straightforward and established statistical techniques to derive labels.

Relative Frequency We simply set $\tilde{p} = r/n$, where n is the number of training examples satisfying α and r the number satisfying $\alpha \wedge \beta$. The reliability of relative frequency as an estimate was measured by *relative cover* (n/N), which is simply the proportion of training examples which satisfy α and hence 'fire' the rule $\alpha \rightarrow \beta$. We use relative cover, since, for example, an estimate of $p = 1$ is more reliable with $r = 100, n = 100$ than with $r = 2, n = 2$ (recall the example in the previous section, which had $r/n = 1$ but test accuracy of only 50%).

Bayesian In Bayesian estimation of probabilities, a prior probability distribution over possible values of p is used. This is then updated to give a posterior

distribution, the mean of which is used as a point estimate $\tilde{p}$ of p. Let λ be the mean of the prior distribution, we then have

$$\tilde{p} = \left(\frac{n}{n+\hat{K}}\right)\frac{r}{n} + \left(\frac{\hat{K}}{n+\hat{K}}\right)\lambda$$

The balance between r/n and λ is governed by the value $\hat{K}$, $\hat{K} = 0$ renders $\tilde{p} = r/n$. Various values for $\hat{K}$ have been employed in the statistical and machine learning literature [Bishop *et al.*, 1975; Cestnik, 1990; Cestnik and Bratko, 1991; Džeroski *et al.*, 1992]. Below we have used the value $\hat{K} = \sqrt{n}$; for the properties of this particular estimate see [Bishop *et al.*, 1975]. The value λ can be seen as a 'guess' at p prior to looking at domain information. We used an estimate of the value $P(\beta)$ for λ, an approach common in the machine learning literature. $P(l < t)$ was calculated by integrating between $\tilde{p} - t$ and $\tilde{p} + t$ on the posterior distribution. The actual value of t is not crucial. It affects the magnitude of $P(l < t)$, but rarely affects the preference ordering. In our experiments t was set to 0.025.

Pseudo-Bayes Like Bayesian, but

$$\hat{K} = \frac{r(n-r)}{(n\lambda - r)^2}$$

Such an approach is *pseudo*-Bayesian because $\hat{K}$, which is usually seen as a prior parameter is a function of r/n—which is a parameter of the training data.

Generating preference orderings in this way provides an alternative to orderings based on specificity (for example [Poole, 1985; Nute, 1988]). In the context of prioritized logics, some of the issues of specificity and accuracy have been considered in [Cussens and Hunter, 1991; Cussens and Hunter, 1993], but there is a clear need to further clarify this relationship by building on more general results relating non-monotonic reasoning and probabilistic inference [Pearl, 1990; Bacchus, 1990; Bacchus *et al.*, 1992].

A preliminary empirical comparison

In our preliminary comparison, we considered two domains. The first was for rules that predict whether a protein residue is part of an alpha-helix. These rules were defined in terms of relative position in a protein and various biochemical parameters. We call this domain the protein domain. The second was for rules that predict the relative activity of drugs. These rules were defined in terms of the structure of the drug and they provided a partial ordering over the degrees of activity of the drugs. We call this domain the drugs domain.

For the protein domain, from a training set of 1778 examples together with background knowledge consisting of 6940 ground literals, Golem generated 100 clauses for the predicate symbol *alpha_helix* and 99 clauses for the predicate symbol *not_alpha_helix* and hence 199 SF clauses for *alpha_helix* and $\neg$*alpha_helix*. For the drugs domain, from a training set of 1762 examples together with background knowledge consisting of 2106 ground literals, Golem generated 23 clauses for the binary predicate *greater_activity* and 24 for the predicate symbol *not_greater_activity*, giving 47 SF rules for *greater_activity* and $\neg$*greater_activity*.

For the protein domain, Table 1 was formed using a test set of 401 *not_alpha_helix* examples and 322 *alpha_helix* examples. For the drugs domain, Table 2 was formed according to a test set of 513 *greater_activity* examples and 513 *not_greater_activity* examples.

Accuracy The key observation from these tables is that if we define accuracy by the ratio correct/(correct + incorrect), as we do in the 'Accuracy' column, then the performance of Prolog is inferior to that of SF logic. Furthermore, the difference in accuracy between the two variants of SF is negligible.

The marked improvement in accuracy of SF logic over Prolog is contingent on the assumption that we can ignore the examples classified as undecided. In other words, in this interpretation, the increased skepticism of the SF logic is not regarded negatively. However, this is only one way of interpreting the undecided category. If accuracy is defined as the percentage of correct examples, as in the 'Correct' column, then SF is markedly less accurate.

Comparing Definitions 1 and 2 When comparing the variants of SF logic, we find, after rounding, that the same results are obtained using Definition 1 for the three different estimation techniques—this is because they all return similar values. Also, since Definition 1 gives a total ordering on the labels, only those examples that are not covered by any rule are undecided. Using the more skeptical Definition 2, we find that accuracy was close or equal to Definition 1 in all cases.

Skepticism and the Protein Domain In the proteins domain, skepticism, as measured by the percentage of undecided examples, increases significantly as we move from Definition 1 to Definition 2. This increase was greatest using relative frequency, since there, the first value of its label, $\tilde{p} = r/n$, which estimates the accuracy of a rule, can be high even if n and consequently n/N is low. Using Definition 2 with relative frequency, a rule is preferred over another if and only if both the first value of its label ($\tilde{p} = r/n$) *and* the second (n/N) are greater than the respective parts of the label of the competing rule. So many rules with high r/n values but low n values will be preferred over competing rules using Definition 1, but not when using Definition 2.

In contrast, for $\hat{K} = \sqrt{n}$ and pseudo-Bayes, values for $\tilde{p}$ substantially higher than the prior mean λ are only possible if n is reasonably high. If n is high then,

	Correct	Incorrect	Undecided	Accuracy
Prolog with *alpha_helix* clauses	58	42	0	58
Prolog with *not_alpha_helix* clauses	59	41	0	59
Definition 1 using relative frequency	53	31	16	63
Definition 2 using relative frequency	45	25	30	64
Definition 1 using $\hat{K} = \sqrt{n}$	53	31	16	63
Definition 2 using $\hat{K} = \sqrt{n}$	48	27	25	64
Definition 1 using pseudo-Bayes	53	31	16	63
Definition 2 using pseudo-Bayes	50	29	21	63

Table 1: The Protein Domain (all values are percentages)

	Correct	Incorrect	Undecided	Accuracy
Prolog with *greater* clauses	79	21	0	79
Prolog with *not_greater* clauses	80	20	0	80
Definition 1 using relative frequency	70	5	25	93
Definition 2 using relative frequency	70	5	25	93
Definition 1 using $\hat{K} = \sqrt{n}$	70	5	25	93
Definition 2 using $\hat{K} = \sqrt{n}$	70	5	25	93
Definition 1 using pseudo-Bayes	70	5	25	93
Definition 2 using pseudo-Bayes	70	5	25	93

Table 2: The Drugs Domain (all values are percentages)

usually, so will be the second part of the label, $P(l < t)$. So in these cases, the second part of the label is usually high when the first part is, which explains the smaller increase in skepticism as we move from Definition 1 to Definition 2.

Skepticism and the Drugs Domain In the drugs domain, we have, after rounding, the same results for both variants of SF logic and all estimates. This is because conflicts between arguments in Γ occurred only for relatively few test examples.

Summary of Empirical Comparison If we allow increased skepticism, the results given here indicate how a richer formalism such as prioritized logics can be used for increased accuracy in reasoning. Furthermore, this shows how a clearer understanding of generating explicit orderings in terms of statistical inference can support this improved capability.

Discussion

It has been widely acknowledged that non-monotonic logics are of critical importance for artificial intelligence, yet there is some dissatisfaction with the rate and nature of progress in the development of non-monotonic logics that address the needs of artificial intelligence.

Many developments in non-monotonic logics have been based on a set of reasoning problems concerning, for example, inheritance, multiple extensions, and cumulativity. Existing non-monotonic logics can be used to capture these problems in an intuitive fashion by appropriate encoding. Yet for the user of these kinds of formalism, it is not clear which is the most appropriate.

We believe that developing non-monotonic logics for artificial intelligence is, in part, an engineering problem and that the space of possible logics that could constitute a solution is enormous. It is therefore necessary to augment theoretical analyses of non-monotonic logics with sound empirical analyses. This should then focus the endeavour on improving the performance of reasoning with uncertain information, and as a matter of course should raise further important and interesting theoretical questions.

Acknowledgements

This work has been funded by UK SERC grants GR/G 29861 GR/G 29878 and GR/G 29854. The authors are grateful for helpful feedback from Dov Gabbay, Donald Gillies and Stephen Muggleton, and also from two anonymous referees.

References

Bacchus, Fahiem; Grove, Adam; Halpern, Joseph Y.; and Koller, Daphne 1992. From statistics to belief. In *Tenth National Conference on Artificial Intelligence (AAAI-92)*. 602–608.

Bacchus, Fahiem 1990. *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach to Probabilities*. MIT Press, Cambridge, MA.

Bishop, Yvonne M. M.; Fienberg, Stephen E.; and Holland, Paul W. 1975. *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, Cambridge, Mass.

Cestnik, Bojan and Bratko, Ivan 1991. On estimating probabilities in tree pruning. In Kodratoff, Yves, editor 1991, *Machine Learning—EWSL-91*. Lecture Notes in Artificial Intelligence 482, Springer-Verlag. 138–150.

Cestnik, Bojan 1990. Estimating probabilities: A crucial task in machine learning. In Aiello, L., editor 1990, *ECAI-90*. Pitman. 147–149.

Cussens, James and Hunter, Anthony 1991. Using defeasible logic for a window on a probabilistic database: some preliminary notes. In Kruse, R. and Seigel, P., editors 1991, *Symbolic and Quantitative Approaches for Uncertainty*. Lecture Notes in Computer Science 548, Springer-Verlag. 146–152.

Cussens, James and Hunter, Anthony 1993. Using maximum entropy in a defeasible logic with probabilistic semantics. In *Information Processing and the Management of Uncertainty in Knowledge-Based Systems (IPMU '92)*. Lecture Notes in Computer Science, Springer-Verlag. Forthcoming.

Cussens, James 1993. Bayes and pseudo-Bayes estimates of conditional probability and their reliability. In *European Conference on Machine Learning (ECML-93)*. Springer-Verlag.

Džeroski, Sašo; Cestnik, Bojan; and Petrovski, Igor 1992. The use of Bayesian probability estimates in rule induction. Turing Institute Research Memorandum TIRM-92-051, The Turing Institute, Glasgow.

Gabbay, Dov and de Queiroz, Ruy 1993. Extending the Curry-Howard interpretation to linear, relevance and other resource logics. *Journal of Symbolic Logic*. Forthcoming.

Gabbay, Dov 1991a. Abduction in labelled deductive systems: A conceptual abstract. In Kruse, R. and Seigel, P., editors 1991a, *Symbolic and Quantitative Approaches for Uncertainty*. Lecture Notes in Computer Science 548, Springer-Verlag. 3–11.

Gabbay, Dov 1991b. Labelled deductive systems. Technical report, Centrum für Informations und Sprachverarbeitung, Universität München.

Hunter, Anthony 1992. A conceptualization of preferences in non-monotonic proof theory. In Pearce, D. and Wagner, G., editors 1992, *Logics in AI*. Lecture Notes in Computer Science 633, Springer-Verlag. 174–188.

Hunter, Anthony 1993. Using priorities in non-monotonic proof theory. Technical report, Imperial College, London. Submitted to the *Journal of Logic, Language, and Information*.

Muggleton, Stephen and Feng, Cao 1990. Efficient induction of logic programs. In *Proc. of the First Conference on Algorithmic Learning Theory*, Tokyo. 473–491.

Muggleton, Stephen 1991. Inductive logic programming. *New Generation Computing* 8:295–318.

Muggleton, Stephen, editor 1992. *Inductive Logic Programming*. Academic Press.

Nute, Donald 1988. Defeasible reasoning and decision support systems. *Decision Support Systems* 4(1):97–110.

Pearl, Judea 1990. Probabilistic semantics for non-monotonic reasoning: A survey. In Shafer, Glen and Pearl, Judea, editors 1990, *Readings in Uncertain Reasoning*. Morgan Kaufmann, San Mateo, CA, USA.

Poole, David L. 1985. On the comparison of theories: Preferring the most specific explanation. In *Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*. 144–147.