# A Classification and Survey of Preference Handling Approaches in Nonmonotonic Reasoning

James Delgrande
School of Computing Science
Simon Fraser University
Burnaby, B.C.
Canada V5A 1S6
jim@cs.sfu.ca

Torsten Schaub*
Institut für Informatik
Universität Potsdam
Postfach 90 03 27
D–14439 Potsdam, Germany
torsten@cs.uni-potsdam.de

Hans Tompits
Institut für Informationssysteme 184/3
Technische Universität Wien
Favoritenstraße 9–11
A–1040 Vienna, Austria
tompits@kr.tuwien.ac.at

Kewen Wang
School of Computing and
Information Technology
Griffith University
Brisbane, QLD 4111, Australia
K.Wang@cit.gu.edu.au

**Abstract**

In recent years, there has been a large amount of disparate work concerning the representation and reasoning with qualitative preferential information by means of approaches to nonmonotonic reasoning. Given the variety of underlying systems, assumptions, motivations, and intuitions, it is difficult to compare or relate one approach with another. Here, we present an overview and classification for approaches to dealing with preference. A set of criteria for classifying approaches is given, followed by a set of desiderata that an approach might be expected to satisfy. A comprehensive set of approaches is subsequently given and classified with respect to these sets of underlying principles.

## 1 Introduction

The notion of *preference* is pervasive in commonsense reasoning, in part because preferences constitute a very natural and effective way of resolving indeterminate situations. In decision making, for example, one may have various desiderata, not all of which can be simultaneously satisfied; in

* Affiliated with the School of Computing Science at Simon Fraser University, Burnaby, Canada.

such a situation, preferences among desiderata may allow one to come to an appropriate compromise solution. In legal reasoning, laws may conflict. Conflicts may be resolved by principles such as ruling that newer laws will have priority over less recent ones, and laws of a higher authority will have priority over laws of a lower authority. For a conflict among these principles, one may further decide that the "authority" preference takes priority over the "recency" preference.

Preference has a decidedly nonmonotonic flavour. Or, more accurately, it may be considered as having a *fundamental* nonmonotonic aspect. Given a preference ordering, however constituted, and some basic or case-specific information, $\Psi$, one may come up with a set of desired outcomes. However, a superset of this case-specific information, $\Psi \cup \Phi$, may lead to a different set of desired outcomes. For example, imagine feeding information into an automated financial advisor: that one is a relatively cautious investor, that one has a long-term horizon, etc. Given these preferences, a set of recommended mutual funds may be suggested by the automated advisor. If the user subsequently states that they also prefer that their funds invest in environmentally and socially responsible companies, then a different set of suggestions may well result.

In logic-based AI, a standard approach to handling preferences is to take an existing system of nonmonotonic reasoning and, in one fashion or another, equip it with preferences. For example, preferences are added in such a manner to default logic [Brewka,1994; Delgrande and Schaub,2000a], autoepistemic logic [Konolige,1988; Rintanen,1994], circumscription [McCarthy,1986; Lifschitz,1985], and logic programming [Zhang and Foo,1997a; Brewka and Eiter,1999]. However, although the notion of "preference" is intuitively straightforward, there is a surprising variety in how this notion is realised in various approaches. Thus, some approaches take a preference ordering as expressing a "desirability" that a property be adopted while in others the ordering expresses the order in which properties (or whatever) are to be considered. As we later describe, some approaches conflate the notion of *inheritance of properties* with the general notion of preference. The outcome of course is that, depending on how the notion of preference is interpreted, different conclusions may be forthcoming. At the same time, while logical preference handling already constitutes an indispensable means for legal reasoning systems (cf. [Gordon,1993; Prakken,1997b]), it is also being used in other application areas such as intelligent agents and e-commerce [Grosof,1999], and the resolution of grammatical ambiguities [Cui and Swift,2001].

In this paper, we survey various approaches to handling qualitative preference information that have appeared in the literature. The intent is to consider ways in which the general notion of *preference* may be interpreted in a system, and to classify and evaluate approaches based on certain criteria. We begin, in the next section, by considering a possible classification of preference approaches. As well, we discuss a number of desiderata that an approach or system may be expected to satisfy. In the following section, we compare and contrast extant systems with respect to these criteria, concentrating on points of interest illustrated by a particular approach. An appendix contains a comprehensive survey of individual approaches.

## 2 Comparing Approaches to Preference

In this section, we consider a number of ways in which approaches to representing and reasoning with preferences can be compared. In the first subsection, we consider ways to *classify* approaches to preference—that is, relatively neutral criteria by which approaches may be distinguished or compared. In the second subsection, we suggest possible *desiderata* for approaches, or properties that an approach ideally will satisfy.[1]

Informally, a preference relation will be a binary relation $<$ between objects of a specific type (formulas, rules, sets of objects, etc.). Most often, $<$ will be a partial order. The idea is that objects with higher precedence or preference are to be asserted (concluded, applied, ...) over lower ranked objects. Thus, for $\delta_2 < \delta_1$, if $\delta_1$ and $\delta_2$ are in conflict, one might expect, all other things being equal, that the higher-ranked object $\delta_1$ will be asserted over the lower, $\delta_2$.[2] Different approaches have further interpreted or constrained the relation $<$ in a multitude of ways; it is the purpose of this paper then to try to provide some framework, or perspective, to these various approaches.

There is one large and important class of preference-like relations that we do not discuss here, that associated with *inheritance of properties*. Essentially, in inheritance of properties, the preference ordering is determined by the *specificity* of antecedent information. As well, with inheritance, one only infers properties from the most specific applicable subclass. Consider rules concerning primary means of locomotion: "animals normally walk", "birds normally fly", "penguins normally swim". If we learn that some object is a penguin (and so a bird and animal), then we would want to apply the highest-ranked default and, all other things being equal, conclude that it swims. However, if the penguin in question is hydrophobic, and so does not swim, we would not want to inherit the next most specific conclusion, that it flies, and so in this case we would conclude nothing about locomotion. However, in a preference ordering, one would try to apply the next default and so, again all other things being equal, conclude that the penguin flies. So inheritance of properties leads to quite different behaviour from preference orderings, as we interpret them here. We refer the reader to [Delgrande and Schaub,2000b] for a full discussion; a survey of approaches to inheritance reasoning is given by Horty [1994].

## 2.1 Classifying Approaches to Preference

We describe here a number of ways in which approaches to preference may be classified. For ease of exposition and concreteness, we will most often use default logic [Reiter,1980][3] to illustrate various concepts. Thus, we may write

$$\frac{:\ red}{red} < \frac{:\ blue}{blue} < \frac{:\ green}{green} \tag{1}$$

to show a preference over colours, implemented as an ordering on default rules. However, it should be emphasised that this is for illustration only; we have no particular preference for default logic. Some other system could be the "host" system as well, preferences need not be on rules, and so on. Similarly, a phrase such as "a higher-ranked rule is applied" is simply an abbreviation for the much more cumbersome "a higher-ranked object (be it a rule, term, formula, set, etc.) is applied (concluded, asserted, etc.)".

[1]Note, however, that the difference between a criterion and desideratum is not necessarily a clear-cut distinction.

[2]Note that some approaches use $<$ in the opposite sense to us.

[3]Very briefly, a default rule $\frac{A:B}{C}$ can be interpreted as "if $A$ is provable and $B$ is consistent with what is believed, then conclude $C$".

We have the following set of not-necessarily independent criteria for classifying approaches to preference:

**Host system** Previously (during the 1990's) default logic [Reiter,1980] was by-and-large the host system of choice, in that the majority of approaches to adding preferences added them to default logic. More recently the emphasis has shifted to logic programs, and in particular extended logic programs. Likely this change reflects a general shift in focus in the research community, from default logic being the most popular nonmonotonic reasoning formalism, to the emergence of extended logic programs and answer set programming. The main facet that can be said about the "underlying system" is that it is easier to compare approaches that use the same base system. As well, a specific approach to preference may be "ported" from one underlying system to another, as for example is done in the methods of Delgrande and Schaub [2000a; 2003] and Brewka and Eiter [1999; 2000]. Thus, in the case of [Brewka and Eiter,1999; 2000], an approach to dealing with preferences was first proposed for answer set programming; this was subsequently expressed using default logic as the host system.

**Meta-level vs. object-level preferences** Most commonly, given some underlying host system, a preference ordering is imposed "externally" on rules of the system. For example, a default theory $(D, W)$ may be extended to a *preferred default theory* $(D, W, <)$ where $< \subseteq D \times D$ gives a preference ordering on how rules may be applied. Alternatively, preferences may be imposed at the object-level. For example, in the approach of Delgrande and Schaub [2000a], constants representing names are associated with the default rules. Instead of a relation $\delta_2 < \delta_1$ between default rules, one can now assert $n_2 \prec n_1$ between the corresponding names, where $\prec$ is a (new) binary relation in the object language.

In the first case, for meta-level preferences, the underlying host system is used more or less as a black box by an enveloping preference system. Thus, for a preferred default theory $(D, W, <)$, the standard meta-level approach is to generate the full set of extensions of the underlying default theory $(D, W)$, order these extensions according to the relation $<$, and then choose the most preferred extension(s). In contrast, an object-level approach to preference allows the use of preferences within the object theory. Thus, in a default theory $(D, W)$, one could have the symbol $<$ appear among formulas in $W$ and $D$. In the approach of Delgrande and Schaub [2000a], for example, a default theory $(D, W)$ that used the (object-level) symbol $\prec$ to express preferences among named default rules in $D$, is mapped to a second theory $(D', W')$ such that the symbol $\prec$ does not appear in $(D', W')$, yet the extensions of $(D', W')$ correspond in a precise sense to the $\prec$-maximal extensions of $(D, W)$.

External, or meta-level preferences, have the advantage that they are (usually) easier to realise: either the underlying inference relation is modified to take into account preferences, or else the underlying approach is used as a black box by a higher-level preference system. On the other hand, the object-level approach allows one to formalise preferences *within* a theory, instead of *about* a theory. As well, the object-level approach is more flexible. For example one may cancel preferences or apply preferences in a context; thus $A \supset \{\neg\}(n_2 \prec n_1)$ could be used to assert that if $A$ is true then it is (not) the case that $n_2 \prec n_1$. As a second example, one could have preferences apply by default; thus, the rule

$$\frac{A : n_2 \prec n_1}{n_2 \prec n_1}$$

asserts that if $A$ is true, then by default we have the preference $n_2 \prec n_1$.

**Static vs. dynamic preferences** A closely-related distinction to the preceding concerns whether preferences are *static*, or fixed at the time the theory is specified, or *dynamic*, and so can be determined "on the fly". An approach with external preferences will, of necessity, have static preferences. For object-level preferences, these preferences could be either static or dynamic. Consider default logic: an approach with static, object-level preferences would have preferences appearing only in the world knowledge $W$. Otherwise preferences appearing in $D$ would be (potentially) dynamic, since their applicability would be determined in the course of determining an extension of the theory. In the case of extended logic programs, an approach with static, object-level preferences, would have preferences appearing only as ground facts (i.e., as rules of the form $(n_2 \prec n_1) \leftarrow$).

**Properties of the preference ordering** The majority of approaches assume that the relation $<$ is a (irreflexive) partial order; this seems to be the minimal notion that would justify the use of the term "preference". However, one might go on and impose further conditions, such as connectivity or (in the case of infinite orderings) well-foundedness. Some approaches assume that $<$ is a total order, since orderings of this kind are easier to deal with. As an intermediate approach between these two possibilities, as we describe subsequently, in some approaches $<$ is assumed to be a partial order, but this is extended to a set of total orders before applying the preference relation.

**What is the preference ordering an ordering on?** Obviously, a preference ordering $<$ is a binary relation on objects of some given type. An issue then concerns the *specific kind* of objects that $<$ is a binary relation on. Although seemingly clear-cut, there are some subtleties here.

First, in default logic or extended logic programs, preferences would naturally be on the rules in a theory. However, we have already noted one distinction: in an external (meta-level) preference relation, the preferences are indeed on the rules themselves. In an object-level preference relation, the preferences are expressed on constants naming the rules; it is then up to the implementer of such an approach to ensure that these constants do indeed denote the rules in question.

Second, there is a distinction between what a user would regard as a preference, and how the preference would be implemented. Thus, it makes sense to think informally of preferences as being on formulas: for example, one might wish to express that green things are preferred to blue things, which are preferred to red. This could be expressed within a first-order language by predicates such as $\mathit{pref}(\mathit{green}(x), \mathit{blue}(x))$ and $\mathit{pref}(\mathit{blue}(x), \mathit{red}(x))$. Thus preferences would be expressed on (reified) formulas such as $\mathit{green}(x)$. However, for implementation, such a preference relation might be translated into an existing approach. Given the fundamental nonmonotonic character of reasoning with preferences, the "target" system would be an existing approach to nonmonotonic reasoning, itself equipped with a notion of preference. Hence, the preferences above might be translated into a suitably-quantified version of something along the lines of (1). So, the underlying reasoning machinery might make use of (in this case) default logic.

Such a general approach has a number of advantages, including adherence to a knowledge engineering principle that says a user should only be given the power that they need for expressing a problem. As well, here the preference relation $\mathit{pref}(\cdot,\cdot)$ would be translated into preferences on *normal* defaults which might then come with improved complexity characteristics over preferences on general rules. However, the specification of such a "knowledge engineered" language remains largely for future research.

For default logic and extended logic programs, preference one way or another are generally expressed on the rules. Exceptions to this include [Sakama and Inoue,2000], wherein preferences are given directly on *atoms* of the language, along with others such as [Pradhan and Minker,1996; Lifschitz,1985]. As well, we note that for a general approach to preference, an account of preference on *sets* of objects will be needed. For example, in purchasing a car, one might wish to express that a car that is safe and economical is preferred to one that is just safe, which in turn is preferred to one that is safe and powerful. Thus, perhaps:

$$\left\{\frac{:s}{s},\frac{:p}{p}\right\} < \left\{\frac{:s}{s}\right\} < \left\{\frac{:s}{s},\frac{:e}{e}\right\}.$$

The idea of the above theory is that all the defaults in a set of rules must be applicable before the set itself is considered to be applied.

**Prescriptive vs. descriptive preferences** The intuition behind a preference ordering is that higher-ranked rules are to be applied before lower-ranked ones. A major distinction as to how this can be done concerns whether $<$ specifies the order in which defaults are to be applied, or provides a notion of "desirability" that a rule be applied. Consider again default logic: In a *prescriptive* interpretation, the ordering $<$ specifies the order in which defaults are to be considered for application. Thus, one applies (if possible) the most preferred default(s), the next most preferred, and so on. In a *descriptive* interpretation, the preference order represents a ranking on desired outcomes: the desirable (or: preferred) situation is one where the most preferred default(s) are applied.[4] The distinction between these interpretations is illustrated in the following example [Brewka and Eiter,2000]:

$$\frac{:a}{a} < \frac{:\neg b}{\neg b} < \frac{a:b}{b}. \tag{2}$$

Assume that there is no initial world knowledge. In a prescriptive interpretation, one would fail to apply the most preferred default (viz. $\frac{a:b}{b}$) since the antecedent is not provable. However, one might expect to apply the two lesser-preferred defaults, giving an extension containing $\{a,\neg b\}$.[5] In a descriptive interpretation, one might observe that by applying the least-preferred default, the most preferred default can be applied; this yields an extension containing $\{a,b\}$.

[4]This is not necessarily a cut-and-dried distinction; for example, [Brewka and Eiter,2000] contains elements of both.

[5]This is obtained, for instance, in the approaches of Baader and Hollunder [1993a], Brewka [1994], and Marek and Truszczyński [1993]; the approach of Delgrande and Schaub [2000a] yields no "preferred" extension.

A full discussion of this distinction is given by Delgrande and Schaub [2000a]. We briefly recapitulate two salient points here. First, a descriptive interpretation relies on a "global" view of the preference relation $<$, in that lower-ranked rules may be applied in order to allow the application of higher-ranked rules. This in turn implies that a descriptive interpretation is most easily implemented by a meta-level approach, for example, in terms of how well a (standard) extension satisfies the preference ranking. In contrast, with a prescriptive object-level approach, we can potentially axiomatise within a theory how different preference orders interact; see [Delgrande and Schaub,2000a] for details.

Second, a prescriptive interpretation comes, in some sense, with more representational force, and allows a tighter characterisation of a domain. That is, a prescriptive interpretation forces a knowledge-base designer to be explicit about what things should be applied in what order. A descriptive interpretation, on the other hand, is more declarative, in the sense that it gives a wish list of preferences which may or may not be meaningful. This is illustrated by the example in (2), where the default $\frac{a:b}{b}$ has highest priority, but this default can only be applied if the prerequisite is proved; one way that this can come about is by applying the lower-ranked default $\frac{:a}{a}$. But this in turn implies that $\frac{:a}{a}$ should be considered first and so have higher priority than $\frac{a:b}{b}$. As well, there is no situation in which $\frac{a:b}{b}$ can be applied and $\frac{:a}{a}$ cannot. Thus, the inference structure of default logic would seem to dictate that $\frac{:a}{a}$ not be ranked below $\frac{a:b}{b}$. Yet, this is what the order $<$ in (2) stipulates.

**From preferences to preferred results** Given a theory and a set of (object- or meta-level) preferences, the standard reasoning task is to generate a set of preferred outcomes. In default logic or extended logic programs, a preferred set of outcomes would be part of an extension or answer set. The set of all extensions or answer sets would represent the possible sets of preferred outcomes when there is ambiguity in the underlying theory.

The preceding prescriptive/descriptive distinction represents a broad characterisation of methods that may be employed for defining preferred extensions. With respect to the preference ordering $<$, there are also two different strategies:

1. One may define preferred extensions in terms of total orders extending the given partial order $<$ (as, e.g., done in the approach of Brewka [1994]). Each such total order then is used to generate a preferred extension.

2. One may define preferred extensions *directly* from the ordering $<$, unmediated by implied total orderings.

   This second case has two realisations:

   (a) define preferred extensions as (standard) extensions of the preference-free theory satisfying additional conditions in accord with $<$ (e.g., [Sakama and Inoue,2000]);

   (b) define preferred extensions directly from $<$ without reference to the preference-free theory (e.g., [Delgrande and Schaub,2000a]).

Clearly, the last possibility appears on the surface to be the most appealing, since it involves neither extraneous extensions nor specialisations of the preference ordering. On the other hand, there has been no work (that we are aware of) comparing the adequacy of these broad characterizations either from a formal or a pragmatic viewpoint.[6]

## 2.2 Evaluating Approaches to Preference

This subsection discusses a number of desiderata that an approach may be expected to satisfy. To begin with, Brewka and Eiter [1999] propose two "principles" argued to constitute a minimal requirement for preference handling in rule-based systems. While the principles are formulated with respect to static preferences, the second need not be [Delgrande *et al.*,2003]. These principles are expressed with respect to rule-based systems. Thus, approaches such as default logic and logic programming are most naturally covered by these principles, although they are also applicable, for example, to a circumscriptive abnormality theory with preferences.

**Principle I:** *Let $B_1$ and $B_2$ be two extensions[7] of a prioritised theory $(T, <)$ generated by rules $R \cup \{\delta_1\}$ and $R \cup \{\delta_2\}$, where rules $\delta_1, \delta_2 \notin R$. If $\delta_1$ is preferred over $\delta_2$, then $B_2$ is not a preferred extension of $T$.*

The term "generated" is crucial in Principle I: For extension $B$, a rule $\delta$ is a generating rule just if its prerequisites are in $B$ and it is not defeated by $B$.

**Principle II:** *Let $B$ be a preferred extension of a prioritised theory $(T, <)$ and $\delta$ a rule such that at least one prerequisite of $\delta$ is not in $B$. Then, $B$ is a preferred extension of $(T \cup \{\delta\}, <')$ whenever $<'$ agrees with $<$ on priorities among rules in $T$.*

Thus, adding a nonapplicable rule in a preferred extension does not make the extension nonpreferred, so long as prior preferences are not changed.

While Principle I is widely accepted, descriptive approaches do not satisfy Principle II in general. Indeed, a typical descriptive approach violating Principle II is the method proposed by Rintanen [1998b] for adding preferences in default logic. Let us consider the following example taken from [Rintanen,1998b]:

$$\begin{array}{rcl}\delta_1 & = & \frac{a\,:\,b}{b};\\ \delta_2 & = & \frac{:\,\neg a}{\neg a};\\ \delta_3 & = & \frac{:\,a}{a}.\end{array}$$

Here, $E = Th(\{\neg a\})$ is a preferred extension of $(\{\delta_2, \delta_3\}, \{\delta_3 < \delta_2\})$ under Rintanen's semantics and $\delta_1$ is inapplicable with respect to $E$. However, $E = Th(\{\neg a\})$ is not a preferred extension of $(\{\delta_1, \delta_2, \delta_3\}, \{\delta_3 < \delta_2, \delta_2 < \delta_1\})$, and thus Principle II is violated.

[6]This is not totally accurate, since the complexity of various decision problems is known for the major approaches. However, even if we make the eminently reasonable assumption that complexity reflects expressibility, this still says nothing about practical issues.

[7]We prefer the term "extension" to "belief set" as used by Brewka and Eiter [1999]. In using "extension" we do not presuppose anything about the underlying system.

**Complexity** For the major approaches to nonmonotonic reasoning, the complexity of general decision problems of interest is known. One might argue that adding preferences to a given approach should not change the worst-case complexity of a given problem, in the sense that the corresponding versions of these problems, specified with and without preferences, respectively, reside in the same complexity class. Thus, consider a decision problem such as:

*Is $C$ a member of all extensions of theory $T$?*

Arguably, it would be desirable that the overall worst-case complexity does not change if "all extensions" is replaced by "all preferred extensions". The intuition is that if the complexity does change, then substantial additional machinery has been added to the underlying formalism in order to implement preferences.

However, one might argue the opposite. Thus, a possible counterargument is that the introduction of preferences enhances the descriptive power of an existing approach, and so it should come as no surprise that the addition of preferences increases the overall worst-case complexity.

It is interesting to note that, of all the approaches that we have identified as being clearly descriptive (viz. [Rintanen,1998b; Sakama and Inoue,2000; Inoue and Sakama,1999; Ryan,1992]; see also the appendix to this paper), all have higher complexity than their host system. Informally, this may be explained by the fact that descriptive approaches to preference are inherently global in nature, in that the application of a (preferred) rule may depend on the application of other, possibly less preferred, rules. A prescriptive approach, on the other hand, can often be regarded as restricting the order of rule application so to accord with the given preference relation.

**Properties of the host system with and without preferences** For another desideratum, in adding preferences to an approach, the original approach should be changed "minimally" in that, by and large, properties of the approach (at least those unrelated to notions of preference) should remain unaffected. This leads to two specific subcriteria.

- **Is a preferred extension an extension of the theory without preferences?** Thus in a default theory with static preferences $(D, W, <)$, one might expect that an extension of this theory also be an extension of the theory without preferences $(D, W)$. For a circumscriptive abnormality theory with preferences, one might expect that its circumscription implies the circumscription without preferences.

  Similarly, in general, a preferred answer set of an extended logic program should be also an answer set of the extended logic program without preference. However, there are some application domains which require modifications of standard extensions, for example, updating logic programs [Eiter *et al.*,2000] and resolving conflicts caused by classical negation [Buccafurri *et al.*,2002]. In addition, if the preference relation $<$ is empty, the reference theory should have the same extensions as the theory without preferences.

- **Do the properties of the original system remain?** This criterion can actually be seen as a collection of criteria: An approach comes with certain formal properties; arguably, the approach with preferences should maintain the same formal properties (unless there is a good reason not to). For example, normal default theories guarantee the existence of extensions. It would seem reasonable that a normal default theory with preferences also guarantee the existence of extensions. In fact, most prescriptive approaches, e.g., [Delgrande and Schaub,2000a], fail to satisfy this, whenever a preference contradicts an intrinsic application order, as in $\frac{:A}{A} < \frac{A:B}{B}$.[8]

## 3 Representative Approaches

In Section 2, we set out some features that a preference semantics might have. In the following, we will review several selected approaches according to their host systems. We assume that the reader is familiar with the standard nonmonotonic formalisms, including default logic, circumscription, and answer set programming.

### 3.1 Default Logic with Preference

A number of approaches have been proposed to add preference in Reiter's default logic. In this subsection, we review some representative approaches.

Concerning work in default logic, Delgrande and Schaub [2000b] have argued that Reiter and Criscuolo [1981], Etherington and Reiter [1983], and Delgrande and Schaub [1994] address property inheritance. In particular, these approaches are based on the idea of resolving conflicts by appeal to specificity information. Hence, for non-conflicting rules, they may produce inappropriate results when used for preferences. For instance, take

$$\frac{a\ :\ b}{b} < \frac{c\ :\ d}{d}$$

along with $W = \{a, c, \neg d\}$. While one expects a single extension containing $b$, the aforecited approaches would replace the first rule either by

$$\frac{a\ :\ b \wedge \neg c}{b} \quad \text{or} \quad \frac{a\ :\ b \wedge (c \supset d)}{b},$$

neither of which would be applicable for providing $b$.

As we explained before, there are *descriptive* and *prescriptive* interpretations for preference. In the former case, one has a "wish list" where the intent is that one way or another the highest-ranked defaults be applied. In the latter case, the ordering reflects the order in which defaults should be applied.

Rintanen [1995] addresses descriptive preference orders in normal default theories (this despite the paper's title and examples, which would indicate that the paper deals with property inheritance). An order on extensions is defined as follows. A default rule $\frac{A:B}{B}$ is *applied* in extension $E$ just if $A, B \in E$. Extension $E$ is preferred over $E'$ iff there is a $\delta \in D$ applied in $E$ but not in

[8]This again suggests that an interesting area for future research would be to address what general principles should govern preferences. Thus, one may be able to disallow, in a principled fashion, a preference such as the preceding.

$E'$ such that if $\delta'$ is preferred over $\delta$ and $\delta'$ is applied in $E'$ then it is also applied in $E$. In this approach, preference on extensions is given in terms of a total order on preferences among rules; consequently, given a partial order on rules, all total orders that preserve the original partial order must be taken into account. Moreover, in principle, all extensions have to be considered, and then the preferred extension is found via additional tests. Consider an ordered theory where we just have the preference $\frac{:B(x)}{B(x)} < \frac{:A(x)}{A(x)}$ along with $\forall x(\neg A(x) \vee \neg B(x))$. Clearly, the number of total orders resulting from this partial order will be exponential in the number of instances of $A$ and $B$. Similarly, the number of extensions in the unordered theory will be exponential in the number of instances.

Rintanen actually introduced several variants of his basic approach in which preference on extensions is specified in somewhat different ways. In particular, he introduced a notion of preference which is defined in terms of *non-defeated* defaults instead of applied ones (a default $\frac{A:B}{B}$ is defeated in an extension $E$ iff $A, \neg B \in E$). As well, the restriction that only normal default rules are considered is dropped in a subsequent paper [Rintanen,1998b].

For prescriptive approaches, Baader and Hollunder [1993a] and Brewka [1994] present prioritised variants of default logic in which the iterative specification of an extension is modified. A default is only applicable at an iteration step if no default with higher priority is applicable. The primary difference between these approaches rests on the number of defaults applicable at each step. While Brewka allows only for applying a single default that is maximal with respect to a total extension of $<$, Baader and Hollunder allow for applying all most preferred defaults at each step.

In contrast, Delgrande and Schaub [1997; 2000a] translate priorities into standard default theories. In this approach, preferences are expressed using an *ordered default theory*, consisting of default rules, world knowledge, and an ordering, reflecting preference, on the default rules. The preferences are assumed to be *prescriptive*, that is specifying the order in which rules must be applied. In this approach, an ordered default theory is transformed into a second, standard (that is, without preferences) default theory wherein the preferences are nonetheless respected, in that defaults are applied in the prescribed order. In an elaboration of the approach, an ordered default theory is allowed, where preference information is specified as part of an overall default theory. Here, one may specify preferences that hold by default, in a particular context, or give preferences among preferences.

## 3.2 Logic Programs with Preference

There have been numerous proposals for expressing preferences in extended logic programs, including [Brewka,1996; Brewka and Eiter,1999; Buccafurri *et al.*,1999; Delgrande *et al.*,2003; Eiter *et al.*,2000; Gelfond and Son,1997; Sakama and Inoue,1996; Wang *et al.*,2000; Zhang and Foo,1997a]. For our purposes, it is enough to view an extended logic program as a syntactically restricted default theory, having an empty background theory, and only rules of the form

$$\frac{L_1 \wedge \cdots \wedge L_m \ : \ \neg L_{m+1}, \ldots, \neg L_n}{L_0}$$

where $L_i$ for $0 \leq i \leq n$ is a literal.

Similarly as for default logic, the approaches in logic programming with preference usually fall into one of two categories. Approaches in one category employ meta-formalisms for characterising "preferred answer sets". For instance, some approaches in this category generate the preferred answer sets by stipulating additional conditions, reflecting the given preference order, on the standard answer sets of the preference-free program, e.g., as done in the method proposed by Brewka and Eiter [1999]. The approaches in the second category translate an ordered logic program into an ordinary logic program such that the preferred answer sets of the former correspond to the answer sets of latter. This method actually provides an implementation of the corresponding preferred answer sets based on extant logic programming systems. The approaches of Delgrande *et al.* [2003] and Gelfond and Son [1997] fall into this category.

The fundamental idea of the approach due to Brewka and Eiter [1999] is to provide a selection function such that, given an ordinary answer set $S$, one can further check if $S$ is "preferred" in a similar way as the case of ordinary answer sets (without preference). However, this cannot be done in a straightforward way with a partial order. So, one has to produce all the total orders that are compatible with the original partial order. The main reason for this is to resolve conflicts caused by multiple answer sets in the presence of preference information. As a consequence, the resulting semantics allows more preferred answer sets than some other approaches (for example [Delgrande *et al.*,2003; Schaub and Wang,2001]). In general, there are prioritised logic programs which have no preferred answer sets, albeit their preference-free versions do possess standard answer sets. For example, if $(\Pi, <)$ is the following ordered program

$$\begin{array}{rcccl} r_1 & = & c & \leftarrow & not\, b\,; \\ r_2 & = & b & \leftarrow & not\ a \end{array} \tag{3}$$

with $r_1 > r_2$, then $\Pi$ has the unique answer set $\{b\}$ but it is not preferred. For this reason, the authors propose another version of preferred answer sets called *weakly preferred answer sets*.

Rather than modifying the Gelfond-Lifschitz transformation, the approach introduced by Schaub and Wang [2001] defines a prioritised version of the immediate consequence operator and thus defines a new answer set semantics for logic programs. The prioritised immediate consequence operator is designed to derive new information by first applying *the most preferred rules* that are applicable at each step of inference, unlike the standard immediate consequence operator which applies *every rule* that is applicable. Moreover, the less preferred rules still have chance to be applied only if they are applicable at later steps. As a result, one obtains a theory of alternating fixed points for logic programs with preference. There are two dimensions of flexibility in this approach since we can define different semantics by different sets of alternating fixed points as well as modifying the preference strategy in the prioritised immediate consequence operator.

The compilation technique of Delgrande, Schaub, and Tompits [2003] has its roots in an approach proposed for default logic [Delgrande and Schaub,1997; 2000a]. The idea of this approach is to select those answer sets of the program that can be generated in an "order preserving way". This allows to enforce the ordering information during the construction of answer sets. In contrast to the default-logic approach of Delgrande and Schaub [1997; 2000a], in which the notion of an order-preserving extension is defined for static preference orderings only, and the encoding of dynamic preferences relies on an additional predicate $\not\prec$, expressing non-preference between two (names of) rules, the framework of Delgrande *et al.* [2003]

is specified right from the beginning for the dynamic case, and static preferences are just a special instance of dynamic ones. Furthermore, Delgrande and Schaub [1997; 2000a] are primarily concerned with a *specific* preference strategy, whilst one of the goals of Delgrande *et al.* [2003] is to demonstrate the flexibility of the framework by providing encodings for *differing preference strategies*. Lastly, in contradistinction to other preference approaches requiring dedicated algorithms, the existence of readily available solvers for the answer set semantics, like dlv [Eiter *et al.*,1997; 1998] or smodels [Niemelä and Simons,1997], makes a realization of this approach straightforward.

Concerning the above described preference approaches, for a given statically ordered logic program $(\Pi, <)$, the following results can be shown [Schaub and Wang,2001]:

1. Every answer set preferred in the sense of [Delgrande *et al.*,2003] is also preferred by [Wang *et al.*,2000; Schaub and Wang,2001].

2. Every answer set preferred in the sense of [Wang *et al.*,2000; Schaub and Wang,2001] is also preferred by [Brewka and Eiter,1999].

3. Every answer set preferred in the sense of [Brewka and Eiter,1999] is an answer set.

In no instance do we obtain the converse. Thus, these preference approaches form a hierarchy of successively-weaker notions of preference.

Although Gelfond and Son [1997] use defaults rather than logic program rules in their approach to handle preference, the host system of their approach is actually logic programs with classical negation and thus we deal with this approach in the current section. Similar to the compilation technique of Delgrande and Schaub [1997; 2000a] and Delgrande *et al.* [2003], Gelfond and Son also specify preference over rules in the object language. Like defeasible logic [Nute,1987], their approach divides rules into two categories (definite rules and default rules) and actually provides a multi-sorted logical language for nonmonotonic reasoning in extended logic programs with preference. In particular, different approaches to preference can be specified by different sets of preference axioms.

Rather than introducing yet another preference semantics, Eiter *et al.* [2003] deal with a method to implement different preference-handling strategies by means of a meta-interpreter technique. In this approach, a meta-interpreter of ordinary extended logic programs is first given and it is further extended for each preference semantics by adding new meta-program rules which specify the preference strategy. Therefore, this approach separates the representation of answer sets and preference strategy. In addition, the meta-program rules in this approach are mainly designed to express iterative recursive definitions. As a result, this approach admits the capturing of different preference semantics for extended logic programs; in particular, those introduced by Brewka and Eiter [1999], Delgrande *et al.* [2003], and Schaub and Wang [2001].

Zhang and Foo [1997a] present an operational semantics for ordered logic programs based on an iterative reduction to standard programs. The approach admits both static and dynamic preferences, in which the dynamic case is reduced to the static one. Interestingly, the semantics has a certain nondeterministic flavour in the sense that an ordered program may be reduced in more than one way to a standard program. As shown by Zhang [2003a], the somewhat elaborate semantics results in a worst-case complexity which lies at the second level of the polynomial hierarchy. Thus, it is intrinsically harder than standard answer set semantics, providing the polynomial hierarchy does not collapse. This semantics has been applied to specifying *updates of logic programs* [Zhang,2003b] (we discuss update semantics which are related to preference methods in Section 3.5 below).

The method of Sakama and Inoue [1996] has the interesting feature that the given preference order is not a relation between (names of) rules, as in the previous frameworks, but represents a relation between *literals*.[9] This relation is used to determine a preference relation on the answer sets of a disjunctive logic program. Intuitively, an answer set $X_1$ is at least as preferable as an answer set $X_2$ iff there are literals $L_1 \in (X_1 \setminus X_2)$ and $L_2 \in (X_2 \setminus X_1)$ such that $L_1$ has at least the priority of $L_2$, and there is no literal $L_2' \in (X_2 \setminus X_1)$ which has strictly higher priority than $L_1$. An answer set $X$ is preferred iff there is no other answer set which is strictly preferred over $X$. The minimality criterion on answer sets makes this approach (presumably) harder than standard answer set semantics for disjunctive logic programs, yielding a worst-case complexity which lies at the third level of the polynomial hierarchy.

Besides the approaches to introducing preferences over rules and literals, Brewka [2002] investigates the problem of specifying preference over disjunction by introducing a new connective called *ordered disjunction*. In Brewka's approach, alternative, ranked options for problem solutions can be represented by ordered disjunctions in rule heads. For example, the ordered disjunction $A \times B$ means: if possible $A$, but if $A$ is not possible then at least $B$. As a result, the disjunction in rule heads is employed to select preferred answer sets of a program.

Buccafurri, Leone, and Rullo [1999] introduced a language for preference handling called *disjunctive ordered logic*, which includes classical negation but not default negation. This language is subsequently extended to include default negation as well [Buccafurri *et al.*,2002]. In their approach, preference is interpreted as the *inheritance* of objects, and conflicts are resolved by favouring more specific rules according to the inheritance hierarchy. The definition of preferred answer sets is given by incorporating preference into the Gelfond-Lifschitz transformation of the original program. Since preference is mainly employed to resolve conflicts caused by classical negation, and a preferred answer set in this approach may not be an ordinary answer set, it is more reasonable to classify it as an *update semantics*. We furthermore note that the language of disjunctive ordered logic has the same complexity as its underlying host language (viz., disjunctive logic programs).

There are also some attempts to extend the well-founded semantics to logic programs with preference [Brewka,1996; Schaub and Wang,2002]. Finally, other preference-based approaches, without the use of default negation, include [Dimopoulos and Kakas,1995; Pradhan and Minker,1996], as well as the framework of defeasible logics [Nute,1987; 1994] and plausible logic [Billington and Rock,2001].

[9]Cf. [Geffner and Pearl,1992] for another approach to preferences on literals.

## 3.3 Autoepistemic Logic and Preference

Autoepistemic logic [Moore,1985] belongs to the family of modal nonmonotonic logics, capturing the behaviour of an ideally rational agent reasoning about its own beliefs. Although being one of the central formal approaches to nonmonotonic reasoning, it currently receives less attention in the AI community, arguably due to a lack of publicly available solvers. In any case, prioritised versions of autoepistemic logic have been defined in the late 1980's and early 1990's.

To begin with, in *hierarchic autoepistemic logic* [Konolige,1988], the introspection ability of an autoepistemic agent is restricted. More specifically, sets of formulas are divided into a number of layers such that, on layer $n$, the formula $L_m A$ ($m < n$) refers to believing $A$ on layer $m$. Conflicts are resolved on the basis that, for each layered set of formulas, only one stable expansion[10] is obtained. It was shown by Przymusinska [1989] that hierarchic autoepistemic logic can be embedded in standard autoepistemic logic.

A somewhat more general preference-handling formalism for autoepistemic logic is put forth by Rintanen [1994], based on similar ideas as his later approach for prioritised default logic [Rintanen,1995]. In his approach, referred to as *prioritised autoepistemic logic,* Rintanen defines a *prioritisation* as a pair $\langle S, < \rangle$, where $S$ is a set of modal formulas and $<$ is a transitive and asymmetric relation over $S$. Intuitively, the elements of $S$ represent those beliefs which are relevant for determining the total beliefs an agent is going to choose on the basis of a given autoepistemic theory, and, as usual, $A < B$ expresses that the agent is more reluctant to accept the belief $A$ than the belief $B$. Formally, given a prioritisation $\langle S, < \rangle$ and a theory $T$, a stable expansion $E$ of $T$ is $\langle S, < \rangle$-preferred iff there is a strict total order $<'$ of $S$ extending $<$ such that for all stable expansions $E'$ of $T$ it holds that $A \in E \setminus E'$ implies that there is some $B$ with $B <' A$ satisfying $B \in E' \setminus E$, for all formulas $A \in S$. Analogously as for Rintanen's default logic approach, prioritised autoepistemic logic represents a decidedly descriptive approach to preference and yields a higher complexity than the underlying host system.

## 3.4 Prioritised Circumscription

Circumscription [McCarthy,1980] is one of the best-known formalisms of nonmonotonic reasoning. *Prioritised circumscription* [McCarthy,1986; Lifschitz,1985] is an alternative way of introducing circumscription by means of an ordering on tuples of predicates satisfying an axiom. To compute prioritised circumscription, Gelfond and Lifschitz [1988] present a compilation of prioritised circumscription into logic programs. The shortcoming of this method is that there is a class of prioritised circumscription which their method cannot treat. To overcome this problem, several researchers provide methods which transform prioritised circumscription into extended logic programs [Chen,1997; Wakaki and Satoh,1997].

We note in passing that Shoham [1987; 1988] introduced a generalisation of circumscription, referred to as the *preferential models* method, in which the minimisation technique, as expressed by the circumscription schema, need not be based on some elaboration of the subset relation; rather, it employs any strict partial order over models. We refer the interested reader to [Makinson,1994] for a survey on this line of research.

[10]Recall that a stable expansion corresponds to an extension in default logic.

## 3.5 Preference and Updating Logic Programs

In recent years, several approaches for updating logic programs with (sets of) rules have been proposed [Alferes *et al.*,1998; 2000; Sakama and Inoue,1999; Zhang and Foo,1997b; 1998; Eiter *et al.*,2002]. In order to resolve conflicts between new and old information, the update information is assigned, in some sense, higher priority over the current knowledge. However, we note that updating and adding preference relations to logic programs are distinct notions. For example, if a given knowledge base consists of a single fact $a \leftarrow$ which is updated by another fact $\neg a \leftarrow$, then a proper update mechanism yields a consistent result having $\{\neg a\}$ as the unique answer set. However, in a preference approach, assigning the rule $\neg a \leftarrow$ a higher preference over $a \leftarrow$ still generally results in an inconsistent program. Furthermore, as illustrated by this example, a distinguished feature of update semantics is that answer sets of the updated program are in general not answer sets of the original program. Consequently, a combination of updates and preferences yields in general an extended framework. In what follows, we discuss update approaches in which explicit preferences are employed. We further note that an implicit (or explicit) preference to new information is also used in methods for belief revision; however, we shall not pursue such approaches here (cf., e.g., [Gärdenfors and Rott,1995] for a survey on belief revision methods).

In the update approach of Eiter *et al.* [2002], following the general paradigm of Alferes *et al.* [1998], update information is given in the form of sequences $(\Pi_1, \ldots, \Pi_n)$ of logic programs, where each $\Pi_i$ is assumed to update the information given by the initial sequence $(\Pi_1, \ldots, \Pi_{i-1})$. Such sequences of programs are given a declarative semantics by means of a syntactic transformation to an *update program*, which is a single extended logic program in an extended language, based on a *causal rejection principle*. The resultant semantics properly generalises the usual answer set semantics for single logic programs. Alternatively, the update semantics can be described in terms of a modified Gelfond-Lifschitz transformation, which results from the standard construction by removal of rejected rules. For capturing the rejection principle, information about rule rejection is explicitly represented at the object level via rejection atoms. It is shown that this update semantics is equivalent to a fragment of inheritance programs proposed by Buccafurri *et al.* [2002].

An integrated framework combining updates and preferences is proposed by Alferes and Pereira [2000], based on *dynamic logic programs* [Alferes *et al.*,1998] and the preference semantics due to Brewka and Eiter [1999]. In this approach, a new language is defined, modeling sequences $(\Pi_1, \ldots, \Pi_n)$ of programs resulting from consecutive updates of an initial program, together with a priority relation among the rules of all successive programs. The priority relation is itself subject to update. This integrated approach is based on the idea that both preferences and updates eliminate rules: preferences eliminate less preferred rules, selecting among the available stable models, and updates eliminate rules overruled by other ones, in order to generate new models. Preferences require a strict partial order on rules, while updates require a linear temporal order, or other distinct linear structures, allowing nevertheless the production of a tree of linear updating sequences.

In this framework, preferences may be enacted on the results of updates, whereas updates may be used for the purpose of changing preferences. Preferences are under this view intended to select further rules after computing the results of updates. Intuitively, starting from the semantics of updates (discarding rejected rules), the semantics of preferences is defined (discarding unpreferred rules) according to the method of Brewka and Eiter [1999] and aiming at a combination of the two.

An integrated semantics is then formulated for both of them.

Zhang and Foo [1997b] describe the update of a knowledge base of ground literals by means of a prioritised logic program over the same language. An update is itself a prioritised logic program over an extended language, consisting of the following elements: (i) a program $\Pi$ containing initial knowledge rules, inertia rules, and update rules; (ii) a naming function $N$ for rules in the resulting program; and (iii) a partial order $<$ containing a pair $r < r'$ for each inertia rule $r$ and each update rule $r'$ in the new program, stating higher priority of inertia rules.

The possible resulting knowledge base is defined on the basis of the answer sets of the resulting program as follows: if it has no answer set, then the updated knowledge base coincides with the initial one; if its answer set is the set of all literals in the extended language (i.e., if we have inconsistency), then the updated knowledge base is given by the set of ground literals of the original language; if it has a consistent answer set, then the updated knowledge base is given by the set of ground literals of the original language such that the corresponding literal in the extended language belongs to the answer set of the resulting program. It is shown that the introduced definitions satisfy the minimal change property with respect to set inclusion.

In a subsequent paper [Zhang and Foo,1998], the problem of updates is addressed if both old and new knowledge is encoded as an extended logic program. The idea in updating the initial program with respect to the new one is to first eliminate contradictory rules from the initial program with respect to the new one, and then to solve conflicts between the remaining rules by means of a suitable prioritised logic program.

## 3.6 Other Approaches to Preference

Abduction is usually defined as inferring the best or most reasonable explanation (or hypothesis) for a given set of facts. It is a form of nonmonotonic reasoning, since explanations which are consistent in a given context may become inconsistent when new information is obtained. Logic-based abduction especially has attracted a great deal of interest, due to progress in logic programming and nonmonotonic reasoning. A logic-based abductive framework can be formalised as $\mathcal{F} = (T, O, H, \vdash)$, where $T$ is the background theory, given by a logical theory (like, e.g., a set of formulas or a logic program), an observation $O$ is a set of atomic formulas, and the set of abducibles $H$ contains possible individual hypotheses. The task of abduction is to find a subset $S$ of $H$ such that $T \cup S \vdash O$ and $T \cup S$ is consistent. Usually, multiple explanations are obtained and thus an important issue in abduction is how to select the most reasonable explanation.

An approach to deal with preferences in the context of abduction is introduced by Eiter and Gottlob [1995]. In this method, the set of abducibles is partitioned into different levels based on preference and thus explanations containing the most preferable hypotheses are selected.

*Priority logic*, due to You, Wang, and Yuan [2001], provides a form of argumentation-based framework with preference, where a theory (or a program) is a pair consisting of a rule set and a priority relation among rules. If a rule $\delta$ has higher priority over another rule $\delta'$, then the application of $\delta$ will block the application of $\delta'$. This interpretation of preference is different from most approaches in logic programming and default logic. In fact, the most difficult issue in adding preference into argumentative reasoning is how to define a reasonable preference among arguments based on preference among rules. Obviously, this problem is not solved either in priority logic. In addition, there is no default negation in priority logic.

## 4 Conclusion

We have presented an overview and classification of approaches to dealing with preference in nonmonotonic reasoning. A set of criteria for classifying approaches is first given, followed by a set of desiderata that an approach might be expected to satisfy. A comprehensive set of approaches is subsequently given and classified in Appendix A with respect to these sets of principles. The intent is to provide some structure on the area, so that seemingly unrelated systems may be compared or related with each other.

**Acknowledgements** The authors would like to thank the anonymous referees for valuable comments which helped to improve the paper. The first author was supported in part by a Canadian NSERC Discovery Grant. The second author was partially supported by the German Science Foundation (DFG) under grant FOR 375/1 and SCHA 550/6, TP C. The collaboration of the second and third author was partially funded by the Information Society Technologies program of the European Commission, Future and Emerging Technologies under the IST-2001-37004 WASP project. The third author was partially supported by the Austrian Science Fund (FWF) under grant N Z29-INF.

## A Appendix: Classification Summary

In this appendix, we provide a brief summary of selected approaches to incorporating preferences in nonmonotonic logics and logic programs. This summary is of necessity incomplete for two reasons:

1. Due to a large number of references, some approaches have not been included. This does not suggest that these approaches are less important, but rather that their general direction or salient features are illustrated by other work.

2. Each approach is summarised by several attributes, primary among them “preference”, “strategy”, “approach”, “complexity”, and “distinguished properties”. In this last attribute, we include adherence to Brewka and Eiter’s two principles. However, it is not known for some approaches whether they satisfy these two principles. As well, on occasion, the complexity level of an approach is not known. In these cases, the corresponding attributes are omitted or incomplete.

Furthermore, the results given below are either directly obtained from the cited references or are obtained from [Brewka and Eiter,1999; Delgrande and Schaub,2000a; Delgrande *et al.*,2003; Rintanen,1998a; 1998b].

## A.1 Preference in Default Logic

**[Baader and Hollunder,1992; 1993b]:**

**Preference:** preference on rules; static preference; strict partial order

**Strategy:** selection function on extensions; prescriptive

**Approach:** meta-level; integrating preference information into the quasi-inductive definition of a default extension

**Complexity:** same level as host system

**Distinguished properties:** (1) each preferred extension is also an extension without preference; (2) Brewka and Eiter's Principle I is violated, while Principle II holds

**[Brewka and Eiter,2000]:**

**Preference:** preference on rules; static preference (plus extension to dynamic case); strict partial order

**Strategy:** selection function on extensions; semi-prescriptive

**Approach:** meta-level; consider all total orderings, each of which is "applied"

**Complexity:** same level as host system

**Distinguished properties:** (1) each preferred extension is also an extension without preference; (2) Brewka and Eiter's Principle I and II are satisfied

**Related work:** extension of [Brewka and Eiter,1999]; Delgrande *et al.* [2000] give translation into standard default logic

**[Delgrande and Schaub,1997; 2000a]:**

**Preference:** preference on rules; dynamic preference; strict partial order

**Strategy:** selection function on extensions; prescriptive

**Approach:** meta-level (compiling an ordered default theory into an ordinary one); apply the preference ordering "directly"

**Complexity:** same level as host system

**Distinguished properties** (1) each preferred extension is also an extension without preference; (2) Brewka and Eiter's Principle I and II are satisfied

**Related work:** [Delgrande *et al.*,2003]

**[Rintanen,1995; 1998b]:**

**Preference:** preference on rules; static preference; strict partial order

**Strategy:** selection function on extensions; descriptive

**Approach:** meta-level; lexicographic comparison of extensions (derive a lexicographic ordering from the total order on defaults); apply the preference ordering "directly"

**Complexity:** higher level than host system

**Distinguished properties:** Brewka and Eiter's Principle I is satisfied, while Principle II is satisfied only by those variants of the approach in which preferred extensions are defined in terms of non-defeated defaults instead of applied ones

## A.2 Preference in Logic Programming

**[Brewka and Eiter,1999]:**

**Host system:** extended logic programs under answer sets

**Strategy:** selection function on answer sets; semi-prescriptive

**Preference:** preference on rules; static preference; strict partial order

**Approach:** meta-level; consider all total orderings, each of which is "applied"

**Complexity:** same level as host system

**Distinguished properties:** (1) each preferred answer set is also a standard answer set; (2) Brewka and Eiter's Principle I and II are satisfied

**Related work:** Delgrande *et al.* [2000; 2003] and Eiter *et al.* [2003] give translations into standard logic programs and implementation

**[Delgrande *et al.*,2003]:**

**Host system:** extended logic programs under answer sets

**Strategy:** selection function on answer sets; prescriptive

**Preference:** preference on rules; dynamic preference; strict partial order

**Approach:** object level (compiling an ordered logic program into an ordinary one); apply the preference ordering "directly"

**Complexity:** same level as host system

**Distinguished properties:** (1) each preferred answer set is also a standard answer set; (2) Brewka and Eiter's Principle I and II are satisfied

**Related work:** [Delgrande and Schaub,1997; 2000a]

**Implementation:** `www.cs.uni-potsdam.de/∼torsten/plp`

**[Grosof,1997]:**

**Host system:** acyclic extended logic programs

**Strategy:** prescriptive

**Preference:** preference on rules; dynamic preference; strict partial order only on stratified logic programs

**Approach:** meta-level; apply the preference ordering "directly"

**Complexity:** same level as host system

**Distinguished properties:** (1) each ordered logic program without recursion has a unique model; (2) Brewka and Eiter's Principle I and II are satisfied

**Related work:** IBM CommonRules project

**Implementation:** `ebusiness.mit.edu/bgrosof/`

**[Schaub and Wang,2001; Wang *et al.*,2000]:**

**Host system:** extended logic programs under answer sets, regular sets, and well-founded model

**Strategy:** selection function on answer sets; prescriptive

**Preference:** preference on rules; static preference; strict partial order

**Approach:** meta-level; apply the preference ordering "directly"; modify the immediate consequence operator; each semantics is defined as a special class of the alternating fixed points

**Complexity:** same level as host system

**Distinguished properties:** (1) each preferred answer set is also a standard answer set; (2) the well-founded model is correct with respect to the preferred answer sets; (3) Brewka and Eiter's Principle I and II are satisfied

**Related work:** [Baader and Hollunder,1992; 1993b; Schaub and Wang,2002]

**Implementation:** `www.cs.uni-potsdam.de/~torsten/plp`

**[Zhang and Foo,1997a]:**

**Host system:** extended logic programs under answer sets

**Strategy:** modified answer sets

**Preference:** preference on rules; dynamic preference; strict partial order

**Approach:** meta-level; program transformation

**Complexity:** higher level than host system

**Distinguished properties:** Brewka and Eiter's Principle I and II are satisfied

**Implementation:** `www.cit.uws.edu.au/~yan/plps.html`

**[Gelfond and Son,1997]:**

**Host system:** logic programs under answer sets

**Strategy:** modified answer sets; prescriptive

**Preference:** preference on rules; dynamic preference, arbitrary order

**Approach:** object level, meta-interpretation; apply the preference ordering "directly"

**Complexity:** same level as host system

**[Sakama and Inoue,2000]:**

**Host system:** extended logic programs (with disjunction) under answer sets

**Strategy:** selection function on answer sets; descriptive

**Preference:** preference on literals;[11] static preference; strict partial order

**Approach:** meta-level; given preference ordering induces a preference relation on extensions

**Complexity:** higher level than host system

**Distinguished properties:** Brewka and Eiter's Principle II is violated, while Principle I holds

**[Buccafurri *et al.*,1996; 1999; 2002; Laenens and Vermeir,1990; Leone and Rossi,1993]:**

**Host system:** ordered logic

**Strategy:** modified answer sets; prescriptive

**Preference:** preference on rules (called *inheritance hierarchy*); static preference; strict partial order

**Approach:** meta-level; apply the preference ordering "directly"

**Complexity:** same level as host system

**[Kakas *et al.*,1994]:**

**Host system:** logic programs without negation as failure (LPwNF); limited form of classical negation

**Strategy:** modified answer sets

**Preference:** preference on rules; strict preference; strict partial order

**Approach:** meta-level; prioritised argumentation; apply the preference ordering "directly"

**Distinguished properties:** LPwNF can characterize default negation

**[Dimopoulos and Kakas,1995]:**

**Host system:** extension of LPwNF

**Strategy:** modified answer sets

**Preference:** preference on rules; static preference; strict partial order

**Approach:** meta-level; prioritised argumentation; apply the preference ordering "directly"

[11]While preference is defined on literals, we can define $r_1 < r_2$ iff $head(r_1) < head(r_2)$, as the authors suggest (where $head(\cdot)$ denotes the head of a logic program rule).

**[Pradhan and Minker,1996]:**

**Host system:** definite logic programs

**Strategy:** modified answer sets

**Preference:** preference on atoms; static preference; strict partial order

**Approach:** meta-level; employ preference to resolve conflicts between different logic programs; apply the preference ordering "directly"

**[Cui and Swift,2001]:**

**Host system:** logic grammars under well-founded model

**Strategy:** prescriptive

**Preference:** preference on rules; dynamic preference; strict partial order

**Approach:** meta-level; apply the preference ordering "directly";

**Complexity:** same level as host system

**Implementation:** `www.cs.sunysb.edu/~tswift/interpreters.html`

**[Brewka,1996]:**

**Host system:** logic programs under well-founded semantics

**Strategy:** prescriptive

**Preference:** preference on rules; dynamic preference; strict partial order

**Approach:** meta-level; apply the preference ordering "directly"

**Complexity:** same level as host system

**[Prakken,1997a]:**

**Host system:** logic programs

**Strategy:** prescriptive

**Preference:** preference on rules; strict partial order

**Approach:** meta-level; argumentation-based; apply the preference ordering "directly"

## A.3 Preference in Autoepistemic Logic

**[Konolige,1988]:**

**Preference:** layered sets of formulas

**Strategy:** descriptive

**Approach:** meta-level; generate all stable expansions layer by layer; apply the preference ordering "directly"

**Distinguished properties:** Przymusinska [1989] gives an embedding into standard autoepistemic logic

**[Rintanen,1994]:**

**Preference:** preference on formulas; static preference; strict partial order

**Strategy:** selection function on extensions; descriptive

**Approach:** meta-level; ordering on formulas induces an ordering on stable expansions; apply the preference ordering via extended total orders

**Complexity:** higher level than host system

## A.4 Prioritised Circumscription

**[Lifschitz,1985]:**

**Host system:** circumscription

**Strategy:** meta-level; preorder (preferences induce strata)

**Preference:** static preference, preference on special-purpose predicates, viz. $ab(\cdot)$ predicates

**Approach:** meta-level; generate all extensions

**Related work:** Chen [1997], Gelfond and Lifschitz [1988], and Wakaki and Satoh [1997] provide compilations from preferred circumscription into logic programs

## A.5 Preference and Updating Logic Programs

**[Alferes and Pereira,2000]:**

**Host system:** dynamic logic programs

**Strategy:** semi-prescriptive

**Preference:** preference on rules; static preferences; strict partial order

**Approach:** meta-level

**Distinguished properties:** extends update mechanism of Alferes *et al.* [1998] by allowing preferences between rules, using the preference approach of Brewka and Eiter [1999]

**[Zhang and Foo,1997b; 1998]:**

**Host system:** extended logic programs

**Strategy:** modified answer sets

**Preference:** preference on rules

**Approach:** meta-level; program transformation

**Distinguished properties:** describes the update of a logic program using the preference approach of Zhang and Foo [1997a]

## A.6 Preference in Other Nonmonotonic Formalisms

**[Inoue and Sakama,1999]:**

**Host system:** abduction

**Strategy:** selection function on minimal explanations; descriptive

**Preference:** static preference, preference on abducibles (literals)

**Approach:** meta-level; consider all extensions; apply the preference ordering "directly"

**Complexity:** higher level than host system

**Related work:** semantics is equivalent to the preferred answer set semantics of Sakama and Inoue [2000]

**[Nute,1987; 1994; Billington,1993; Antoniou *et al.*,2000]:**

**Host system:** defeasible logic

**Strategy:** prescriptive

**Preference:** preference on rules; static preference; arbitrary order

**Approach:** meta-level; integrating preference into resolution procedure

**[You *et al.*,2001]:**

**Host system:** priority logic (prioritised argumentation)

**Strategy:** deriving preference on arguments from rule preference

**Preference:** preference on rules; static preference; arbitrary order

**Approach:** meta-level; consider all acceptable arguments and apply additional tests

**Complexity:** higher level than host system

**Related work:** prioritised argumentation is also studied by Dimopoulos and Kakas [1995] and Prakken [1997a]

**[Ryan,1992]:**

**Host system:** classical logic (ordered theory presentations)

**Strategy:** descriptive

**Preference:** preference on formulas; static preference; strict partial order

**Approach:** meta-level; consider additional tests on models in accord to given order; apply the preference ordering "directly"

**Complexity:** higher level than host system

## References

[Alferes and Pereira, 2000] J. Alferes and L. Pereira. Updates plus Preferences. In M. Aciego, I. de Guzmán, G. Brewka, and L. Pereira, editors, *Proc. 7th European Workshop on Logics in Artificial Intelligence* (*JELIA 2000*), volume 1919 of *Lecture Notes in Computer Science*, pages 345–360. Springer, 2000.

[Alferes *et al.*, 1998] J. Alferes, J. Leite, L. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic Logic Programming. In A. Cohn, L. Schubert, and S. Shapiro, editors, *Proc. 6th Int. Conf. on Principles of Knowledge Representation and Reasoning* (*KR'98*), pages 98–111. Morgan Kaufmann, 1998.

[Alferes *et al.*, 2000] J. Alferes, J. Leite, L. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming*, 45(1–3):43–70, 2000.

[Antoniou *et al.*, 2000] G. Antoniou, D. Billington, G. Governatori, and M. J. Maher. Defeasible logic versus logic programming without negation as failure. *Journal of Logic Programming*, 42(1):47–57, 2000.

[Baader and Hollunder, 1992] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In B. Nebel, C. Rich, and W. Swartout, editors, *Proceedings of the Third International Conference on the Principles of Knowledge Representation and Reasoning*, pages 306–317, Cambridge, MA, October 1992.

[Baader and Hollunder, 1993a] F. Baader and B. Hollunder. How to prefer more specific defaults in terminological default logic. In R. Bajcsy, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 669–674. Morgan Kaufmann Publishers, 1993.

[Baader and Hollunder, 1993b] F. Baader and B. Hollunder. How to prefer more specific defaults in terminological default logic. Technical Report RR-92-58, DFKI, December 1993.

[Billington and Rock, 2001] D. Billington and A. Rock. Propositional plausible logic: Introduction and implementation. *Studia Logica*, 67:243–269, 2001.

[Billington, 1993] D. Billington. Defeasible logic is stable. *Journal of Logic and Computation*, 3(4):379–400, 1993.

[Brewka and Eiter, 1999] G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109(1-2):297–356, 1999.

[Brewka and Eiter, 2000] G. Brewka and T. Eiter. Prioritizing default logic. In St. Hölldobler, editor, *Intellectics and Computational Logic—Papers in Honour of Wolfgang Bibel*, pages 27–45. Kluwer Academic Publishers, 2000.

[Brewka, 1994] G. Brewka. Adding priorities and specificity to default logic. In L. Pereira and D. Pearce, editors, *European Workshop on Logics in Artificial Intelligence (JELIA'94)*, Lecture Notes in Artificial Intelligence, pages 247–260. Springer Verlag, 1994.

[Brewka, 1996] G. Brewka. Well-founded semantics for extended logic programs with dynamic preferences. *Journal of Artificial Intelligence Research*, 4:19–36, 1996.

[Brewka, 2002] Gerhard Brewka. Logic programming with ordered disjunction. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence* (*AAAI/IAAI'02*), pages 100–105. AAAI Press, 2002.

[Buccafurri *et al.*, 1996] F. Buccafurri, N. Leone, and P. Rullo. Stable models and their computation for logic programming with inheritance and true negation. *Journal of Logic Programming*, 27:5–43, 1996.

[Buccafurri *et al.*, 1999] F. Buccafurri, N. Leone, and P. Rullo. Semantics and expressiveness of disjunctive ordered logic. *Annals of Mathematics and Artificial Intelligence*, 25:311–337, 1999.

[Buccafurri *et al.*, 2002] F. Buccafurri, W. Faber, and N. Leone. Disjunctive logic programs with inheritance. *Theory and Practice of Logic Programming*, 2(3):293–321, 2002.

[Chen, 1997] J. Chen. Embedding prioritized circumscription into logic programs. In *Proc. of the 10th International Symposium on Foundations of Intelligent Systems* (*ISMIS'97*), *LNAI 1325*, pages 50–59. Springer-Verlag, 1997.

[Cui and Swift, 2001] B. Cui and T. Swift. Preference logic grammars: Fixed-point semantics and application to data standardization. *Artificial Intelligence*, 138(1-2):117–147, 2001.

[Delgrande and Schaub, 1994] J. Delgrande and T. Schaub. A general approach to specificity in default reasoning. In J. Doyle, P. Torasso, and E. Sandewall, editors, *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning*, pages 146–157. Morgan Kaufmann Publishers, 1994.

[Delgrande and Schaub, 1997] J. Delgrande and T. Schaub. Compiling reasoning with and about preferences into default logic. In M. Pollack, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 168–174. Morgan Kaufmann Publishers, 1997.

[Delgrande and Schaub, 2000a] J. Delgrande and T. Schaub. Expressing preferences in default logic. *Artificial Intelligence*, 123(1-2):41–87, 2000.

[Delgrande and Schaub, 2000b] J.P. Delgrande and T. Schaub. The role of default logic in knowledge representation. In J. Minker, editor, *Logic-Based Artificial Intelligence*, pages 107–126. Kluwer Academic Publishers, 2000.

[Delgrande *et al.*, 2000] J. Delgrande, T. Schaub, and H. Tompits. A compilation of Brewka and Eiter's approach to prioritization. In M. Ojeda-Aciego, I. Guzmán, G. Brewka, and L. Pereira, editors, *Proceedings of the European Workshop on Logics in Artificial Intelligence* (*JELIA 2000*), volume 1919 of *Lecture Notes in Artificial Intelligence*, pages 376–390. Springer-Verlag, 2000.

[Delgrande *et al.*, 2003] J. Delgrande, T. Schaub, and H. Tompits. A framework for compiling preferences in logic programs. *Theory and Practice of Logic Programming*, 3(2):129–187, 2003.

[Dimopoulos and Kakas, 1995] Y. Dimopoulos and C. Kakas. Logic programming without negation as failure. In J. Lloyd, editor, *Proceedings of the International Symposium of Logic Programming*, pages 369–383. The MIT Press, 1995.

[Eiter and Gottlob, 1995] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42:3–42, 1995.

[Eiter *et al.*, 1997] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A Deductive System for Non-monotonic Reasoning. In *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'97*), pages 363–374, 1997.

[Eiter *et al.*, 1998] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. The KR System dlv: Progress Report, Comparisons and Benchmarks. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning* (*KR'98*), pages 406–417, 1998.

[Eiter *et al.*, 2000] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Considerations on updates of logic programs. In *Proceedings of the Seventh European Workshop on Logics in Artificial Intelligence* (*JELIA'2000*), volume 1919 of *Lecture Notes in Artificial Intelligence*, pages 2–20. Springer-Verlag, 2000.

[Eiter *et al.*, 2002] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming*, 2(6):711–767, 2002.

[Eiter *et al.*, 2003] T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Computing preferred answer sets by meta-interpretation in answer set programming. *Theory and Practice of Logic Programming*, 3(4–5):463–498, 2003.

[Etherington and Reiter, 1983] D.W. Etherington and R. Reiter. On inheritance hierarchies with exceptions. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 104–108. Morgan Kaufmann Publishers, 1983.

[Gärdenfors and Rott, 1995] P. Gärdenfors and H. Rott. Belief revision. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 36–132. Oxford University Press, 1995.

[Geffner and Pearl, 1992] H. Geffner and J. Pearl. Conditional entailment: Bridging two approaches to default reasoning. *Artificial Intelligence*, 53(2-3):209–244, 1992.

[Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. Compiling circumscription theories into logic programs. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 455–459. The MIT Press, 1988.

[Gelfond and Son, 1997] M. Gelfond and T. Son. Reasoning with prioritized defaults. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Third International Workshop on Logic Programming and Knowledge Representation*, volume 1471 of *Lecture Notes in Computer Science*, pages 164–223. Springer-Verlag, 1997.

[Gordon, 1993] T. Gordon. *The Pleading Game: An Artificial Intelligence Model of Procedural Justice*. Dissertation, Technische Hochschule Darmstadt, Alexanderstraße 10, D-64283 Darmstadt, Germany, 1993.

[Grosof, 1997] B. Grosof. Prioritized conflict handling for logic programs. In J. Maluszynsk, editor, *Logic Programming: Proceedings of the 1997 International Symposium*, pages 197–211. The MIT Press, 1997.

[Grosof, 1999] B. Grosof. Business rules for electronic commerce. `http://www.research.ibm.com/rules/papers.html`, 1999. IBM Research.

[Horty, 1994] J. Horty. Some direct theories of nonmonotonic inheritance. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 111–187, 1994.

[Inoue and Sakama, 1999] K. Inoue and C. Sakama. Abducing priorities to derive intended conclusions. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 44–49. Morgan Kaufmann Publishers, 1999.

[Kakas *et al.*, 1994] A. Kakas, P. Mancarella, and P. M. Dung. The acceptability semantics for logic programs. In P. Van Hentenryck, editor, *Proceedings of the International Conference on Logic Programming*, pages 504–519. The MIT Press, June 1994.

[Konolige, 1988] K. Konolige. Hierarchic autoepistemic theories for nonmonotonic reasoning. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 439–443. Morgan Kaufmann Publishers, 1988.

[Laenens and Vermeir, 1990] E. Laenens and D. Vermeir. A fixpoint semantics for ordered logic. *Journal of Logic and Computation*, 1(2):159–185, 1990.

[Leone and Rossi, 1993] N. Leone and G. Rossi. Well-founded semantics and stratification for ordered logic programs. *New Generation Computing*, 12(1):91–121, 1993.

[Lifschitz, 1985] V. Lifschitz. Closed-world databases and circumscription. *Artificial Intelligence*, 27:229–235, 1985.

[Makinson, 1994] D. Makinson. General patterns in nonmonotonic reasoning. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 35–110. Oxford University Press, 1994.

[Marek and Truszczyński, 1993] V. Marek and M. Truszczyński. *Nonmonotonic logic: context-dependent reasoning*. Artifical Intelligence. Springer-Verlag, 1993.

[McCarthy, 1980] J. McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McCarthy, 1986] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[Moore, 1985] R.C. Moore. Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25:75–94, 1985.

[Niemelä and Simons, 1997] I. Niemelä and P. Simons. Smodels: An Implementation of the Stable Model and Well-Founded Semantics for Normal Logic Programs. In *Proceedings of the 4th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR-97*), pages 420–429, 1997.

[Nute, 1987] D. Nute. Defeasible reasoning. In *Proceedings of the 20th Hawaii International Conference on Systems Science*, pages 470–477. IEEE Press, 1987.

[Nute, 1994] D. Nute. Defeasible logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3,* pages 353–395. Oxford University Press, 1994.

[Pradhan and Minker, 1996] S. Pradhan and J. Minker. Using priorities to combine knowledge bases. *International Journal of Cooperative Information Systems*, 5(2-3):333–364, 1996.

[Prakken, 1997a] H. Prakken. Argument-based logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 7:25–75, 1997.

[Prakken, 1997b] H. Prakken. *Logical Tools for Modelling Legal Argument*. Kluwer Academic Publishers, 1997.

[Przymusinska, 1989] H. Przymusinska. The embeddability of hierarchic autoepistemic logic in autoepistemic logic. In *Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems* (*ISMIS'89*), pages 485–493. North-Holland, 1989.

[Reiter and Criscuolo, 1981] R. Reiter and G. Criscuolo. On interacting defaults. In *Proceedings of the International Joint Conference on Artificial Intelligence,* pages 270–276, Vancouver, B.C., 1981.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Rintanen, 1994] J. Rintanen. Prioritized autoepistemic logic. In C. MacNish, D. Pearce, and L. M. Pereira, editors, *Proceedings of the European Workshop on Logics in Artificial Intelligence*, volume 838 of *Lecture Notes in Artificial Intelligence*, pages 232–246, Berlin, September 1994. Springer-Verlag.

[Rintanen, 1995] J. Rintanen. On specificity in default logic. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1474–1479, Montreal, 1995.

[Rintanen, 1998a] J. Rintanen. Complexity of prioritized default logics. *Journal of Artificial Intelligence Research*, 9:423–461, 1998.

[Rintanen, 1998b] J. Rintanen. Lexicographic priorities in default logic. *Artificial Intelligence*, 106:221–265, 1998.

[Ryan, 1992] M. Ryan. Representing defaults as sentences with reduced priority. In B. Nebel, C. Rich, and W. Swartout, editors, *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning*, pages 649–660. Morgan Kaufmann Publishers, 1992.

[Sakama and Inoue, 1996] C. Sakama and K. Inoue. Representing priorities in logic programs. In M. Maher, editor, *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming*, pages 82–96, Cambridge, 1996. The MIT Press.

[Sakama and Inoue, 1999] C. Sakama and K. Inoue. Updating extended logic programs through abduction. In M. Gelfond, N. Leone, and G. Pfeifer, editors, *Proc. 5th Int. Conf. on Logic Programming and Nonmonotonic Reasoning* (*LPNMR'99*), volume 1730 of *Lecture Notes in Artificial Intelligence*, pages 147–161. Springer, 1999.

[Sakama and Inoue, 2000] C. Sakama and K. Inoue. Prioritized logic programming and its application to commonsense reasoning. *Artificial Intelligence*, 123(1-2):185–222, 2000.

[Schaub and Wang, 2001] T. Schaub and K. Wang. A comparative study of logic programs with preference. In B. Nebel, editor, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 597–602. Morgan Kaufmann Publishers, 2001.

[Schaub and Wang, 2002] T. Schaub and K. Wang. Preferred well-founded semantics for logic programming by alternating fixpoints: Preliminary report. In *Proceedings of the Ninth Workshop on Non-Monotonic Reasoning* (*NMR'02*), page `http://arxiv.org/abs/cs.AI/0207060`, 2002.

[Shoham, 1987] Y. Shoham. A semantical approach to non-monotonic logics. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 388–392, 1987.

[Shoham, 1988] Y. Shoham. *Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence*. The MIT Press, Cambridge, MA, 1988.

[Wakaki and Satoh, 1997] T. Wakaki and K. Satoh. Compiling circumscription into extended logic programs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 182–187. Morgan Kaufmann Publishers, 1997.

[Wang *et al.*, 2000] K. Wang, L. Zhou, and F. Lin. Alternating fixpoint theory for logic programs with priority. In *Proceedings of the First International Conference on Computational Logic*, volume 1861 of *Lecture Notes in Computer Science*, pages 164–178. Springer-Verlag, 2000.

[You *et al.*, 2001] J. You, X. Wang, and L. Yuan. Nonmonotonic reasoning as prioritized argumentation. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):968–979, 2001.

[Zhang and Foo, 1997a] Y. Zhang and N. Foo. Answer sets for prioritized logic programs. In J. Maluszynski, editor, *Proceedings of the International Symposium on Logic Programming* (*ILPS'97*), pages 69–84. MIT Press, 1997.

[Zhang and Foo, 1997b] Y. Zhang and N. Foo. Towards Generalized Rule-based Updates. In *Proc. 15th Int. Joint Conf. on Artificial Intelligence* (*IJCAI'97*), volume 1, pages 82–88. Morgan Kaufmann, 1997.

[Zhang and Foo, 1998] Y. Zhang and N. Foo. Updating Logic Programs. In H. Prade, editor, *Proc. 13th Europ. Conf. on Artificial Intelligence* (*ECAI'98*), pages 403–407. Wiley, 1998.

[Zhang, 2003a] Y. Zhang. Logic program based updates. `http://www.cit.uws.edu.au/˜yan/`, 2003. Draft.

[Zhang, 2003b] Y. Zhang. Two results for prioritized logic programs. *Theory and Practice of Logic Programming*, 3(2):223–242, 2003.