The Knowledge Engineering Review, Vol. 13:3, 1998, 225–246
Printed in the United Kingdom

A tutorial on default reasoning

GRIGORIS ANTONIOU

School of Computing & Information Technology, Griffith University, QLD 4111, Australia (e-mail: ga@cit.gu.edu.au)

Abstract

Default reasoning is concerned with making inferences in cases where the information at hand is incomplete. In such cases it is necessary to make plausible assumptions, which in default reasoning are based on default rules. This paper gives an introduction to the field. It discusses in depth one particular approach, default logic, including properties, semantics and computational models. It also gives an overview of other ideas and approaches.

1 Introduction: Default reasoning

When an intelligent system (either computer-based or human) tries to solve a problem, it may be able to rely on complete information about this problem, and its main task is to draw the correct conclusions using classical reasoning. In such cases, classical reasoning methods such as predicate logic may be sufficient.

However, in many situations the system has only *incomplete information* at hand, be it because some pieces of information are unavailable, or because it has to respond quickly and does not have the time to collect all relevant data. In these cases, the system has to make some plausible conjectures, which in the case of default reasoning are based on rules of thumb, called *defaults*. For example, an emergency doctor has to make some conjectures about the most probable causes of the symptoms observed. Obviously, it would be inappropriate to await the results of possibly extensive and time-consuming tests before beginning with the treatment.

When decisions are based on assumptions, these may turn out to be wrong in the face of additional information that becomes available; for example, medical tests may lead to a modified diagnosis. The phenomenon of having to take back some previous conclusions is called *nonmonotonicity*; it says that if a statement φ follows from a set of premises M and $M \subseteq M'$, φ does not necessarily follow from M' (McCarthy, 1980; Reiter, 1980; Marek & Truszczynski, 1993; Antoniou, 1997).

Default logic (Reiter, 1980) is perhaps the most prominent method for nonmonotonic reasoning, basically because of the simplicity of the notion of a default, and because defaults prevail in many application areas. However, there exist several alternative design decisions which have led to variations of the initial idea; actually, we can talk of a *family of default reasoning methods* because they share the same foundations.

In this paper, we present the motivations and basic ideas of some important default reasoning approaches, and compare them both with respect to interconnections and the fulfillment of some properties. When designing a tutorial, it is good to have some aims against which the final product can be tested. The particular aims of this tutorial paper are the following:

- To present the basic ideas of default logic to persons without prior knowledge in the area. Only basic understanding of classical logic is required.
- To equip readers with skills and methods so that they can apply the concepts of default logic to concrete situations.

- To give the reader a feeling of the diversity of the topic.

The paper is organised as follows. Section 2 presents the basics of Reiter's default logic. It includes the definition of the basic concepts, a computational model, and discussion of important properties. Section 3 gives an overview of several default logic approaches. Section 4 provides two semantic characterisations of default logic: one based on argumentation, and one based on possible worlds. Section 5 briefly discusses two default reasoning approaches based on logic programming: the stable model semantics of logic programs, and Nute's Defeasible Logic (Nute, 1987, 1994). Finally, section 6 outlines some applications of default reasoning.

2 Default logic

2.1 The notion of a default

Suppose I am asked in the morning how I intend to go to work. My answer is "By bus", because usually I go to work by bus. This rule of thumb is represented by the default

$$\frac{goToWork : useBus}{useBus}$$

which is read as follows: "If I have to go to work, and if I may assume that I can use the bus (meaning that nothing to the contrary is known), then I decide to use the bus". So I leave home, walk to the bus station and read a notice saying "No buses today because we are on strike!". Now I definitely know that I cannot use the bus to go to work, therefore the default is no longer applicable (now I cannot assume that I may use the bus). I have to take back my previous conclusion, so my behaviour is nonmonotonic.

Before proceeding with more examples, let us first explain why classical logic is not appropriate to model this situation. Of course, I could use the rule

$$goToWork \wedge : strike \; \omega \quad useBus$$

Except that this rule is still insufficient: another reason for not wanting to use the bus is that I am late and therefore in a hurry. Then the rule could be modified as follows:

$$goToWork \wedge : strike \wedge : inHurry \; \omega \quad useBus$$

But there are more conceivable reasons for my not using a bus, for example icy streets. A first difficulty of a predicate logic solution becomes apparent: we would have to list *all* possible obstacles, however unprobable they may be. Is this possible, especially in a rapidly changing environment?

But the real difficulty actually lies in the use of the rule: I would have to definitively establish that all the preconditions of the rule are true (that is that the obstacles are not given) before I could apply the rule. So every morning I would have to call news agencies, listen to the radio etc. to obtain all the information needed about weather, strikes, etc., before making up my mind. It might be afternoon when I would finally arrive at work, only to be told that I am fired because I missed giving my lecture.

Let us now turn to more serious examples. Defaults can be used to model *prototypical reasoning*, which means that most instances of a concept have some property. One example is the statement "Typically, children have (living) parents", which may be expressed by the default

$$\frac{child(X) : hasParents(X)}{hasParents(X)}$$

A further form of default reasoning is *no-risk reasoning*. It concerns situations where we draw a conclusion even if it is not the most probable, because another decision could lead to a disaster. Perhaps the best example is the following main principle of justice in the Western cultures: "In the absence of evidence to the contrary assume that the accused is innocent". In default form:

$$\frac{accused(X) : innocent(X)}{innocent(X)}$$

Defaults naturally occur in many application domains. Let us give an example from legal reasoning. According to German law, a foreigner is usually expelled if they have committed a crime. One of the exceptions to this rule concerns political refugees. This information is expressed by the default

$$\frac{criminal(X) \wedge foreigner(X) : expel(X)}{expel(X)}$$

in combination with the rule

$$politicalRefugee(X)\ \omega\ :\ expel(X)$$

Defaults can be used naturally to model the *Closed World Assumption* (Reiter, 1978) which is used in database theory, algebraic specification, and logic programming. According to this assumption, an application domain is described by certain axioms (in the form of relational facts, equations, rules etc.) with the following understanding: a ground fact (that is, a non-parameterised statement about single objects) is taken to be false in the problem domain if it does not follow from the axioms. The closed world assumption has the simple default representation

$$\frac{true : : \varphi}{: \varphi}$$

for each ground atom φ. The explanation of the default is: if it is consistent to assume : φ (which is equivalent to not having a proof for φ) then conclude : φ.

Further examples of defaults can be found in, say, Besnard (1989), Etherington (1987b), Lukaszewicz (1990) and Poole (1994).

2.2 Logical notation

This paper does not require knowledge of logic other than an understanding of basic logical notions. The notation used is rather standard. The symbols : , $\frown$, $\wedge$, ω denote the logical connectives "not", "or", "and" and "implies". As is standard in predicate logic, $\varphi\ \omega\ \psi$ (φ implies ψ) is equivalent to : $\varphi \frown \psi$.

In a given logical language, *For* denotes the set of all formulae in the language. The symbols ℓ and $\models$ denote syntactic and semantic consequence, respectively. Finally, for a set of formulae M, $Th(M)$ denotes the deductive closure of M. In case $M = Th(M)$ then M is called deductively closed.

2.3 The syntax of default logic

A *default theory* T is a pair (W, D) consisting of a set W of predicate logic formulae (called the *facts* or *axioms* of T) and a countable set D of defaults. A *default* δ has the form

$$\frac{\varphi : \psi_1, \triangleright\triangleright\triangleright, \psi_n}{\chi}$$

where $\varphi, \psi_1, \triangleright\triangleright\triangleright, \psi_n, \chi$ are closed predicate logic formulae[1], and $n > 1$. The formula φ is called the *prerequisite*, $\psi_1, \triangleright\triangleright\triangleright, \psi_n$ the *justifications*, and χ the *consequent* of δ. Sometimes φ is denoted by $pre(\delta)$, $f\psi_1, \triangleright\triangleright\triangleright, \psi_n g$ by $just(\delta)$, and χ by $cons(\delta)$. For a set D of defaults, $cons(D)$ denotes the set of consequents of the defaults in D. A default is called *normal* iff it has the form $(\varphi : \psi)/\psi$.

One point that needs some discussion is the requirement that the formulae in a default be ground. In the following, whenever we use defaults with free variables, we interpret them as schemata representing the set of all possible ground instances in the logical language of the default theory T.

[1] That means, they have no free variables.

2.4 Operational semantics: Extensions

Given a default $(\varphi : \psi_1, \triangleright\triangleright\triangleright, \psi_n)/\chi$, its informal meaning is the following: *If φ is known, and if it is consistent to assume $\psi_1, \triangleright\triangleright\triangleright, \psi_n$, then conclude χ.* If E is a set of formulae which represents the current knowledge base, that is, the information currently available, then we can formalize the above intuition as follows:

$$\delta = \frac{\varphi : \psi_1, \triangleright\triangleright\triangleright, \psi_n}{\chi}$$

is *applicable to* a deductively closed set of formulae E iff $\varphi \in E$ and : $\psi_1 \notin E, \triangleright\triangleright\triangleright, : \psi_n \notin E$.

The meaning of a knowledge base in default logic (a default theory) is given in terms of *extensions* which, intuitively, represent possible consistent interpretations of the available information. A default theory can have more than one extension, as we will see. This happens when default rules support contradictory conclusions.

For a given default theory $T = (W, D)$ let $\Pi = (\delta_0, \delta_1, \triangleright\triangleright\triangleright)$ be a finite or infinite sequence of defaults from D without multiple occurrences. Think of Π as a possible order in which we apply some defaults from D. Of course, we don't want to apply a default more than once within such a reasoning chain because no additional information would be gained by doing so. We denote the initial segment of Π of length k by $\Pi[k]$, provided the length of Π is at least k (from now on, this assumption is always made when referring to $\Pi[k]$). With each such sequence Π we associate two sets of first-order formulae, $In(\Pi)$ and $Out(\Pi)$:

- $In(\Pi)$ is $Th(W \cup \{fcons(\delta) \mid \delta$ occurs in $\Pi\})$. So, $In(\Pi)$ collects the information gained by the application of the defaults in Π and represents the *current knowledge base* after the defaults in Π have been applied.
- $Out(\Pi) = \{\neg \psi \mid \psi \in just(\delta)$ for some δ occurring in $\Pi\}$. So, $Out(\Pi)$ collects formulae that should not turn out to be true, i.e. that should not become part of the current knowledge base even after subsequent application of other defaults.

Let us give a simple example. Consider the default theory $T = (W, D)$ with $W = \{a\}$ and D containing the following defaults:

$$\delta_1 = \frac{a : : b}{: b}, \quad \delta_2 = \frac{b : c}{c}$$

For $\Pi = (\delta_1)$ we have $In(\Pi) = Th(\{a, : b\})$ and $Out(\Pi) = \{b\}$. For $\Pi = (\delta_2, \delta_1)$ we have $In(\Pi) = Th(\{a, c, : b\})$ and $Out(\Pi) = \{\neg : c, b\}$.

Up to now, we have not assured that the defaults in Π can be applied in the order given. In the example above, (δ_2, δ_1) cannot be applied in this order (applied according to the definition in the previous subsection). To be more specific, δ_2 cannot be applied, since $b \notin In(()) = Th(W) = Th(\{a\})$ which is the current knowledge before we attempt to apply δ_2. On the other hand, there is no problem with $\Pi = (\delta_1)$; in this case we say that Π is a *process of T*. Here is the formal definition:

- Π is called a *process of T* iff δ_k is applicable to $In(\Pi[k])$, for every k such that δ_k occurs in Π.

Given a process Π of T we define the following:

- Π is *successful* iff $In(\Pi) \cap Out(\Pi) = \{, , \}$, otherwise it is *failed*.
- Π is *closed* iff every $\delta \in D$ that is applicable to $In(\Pi)$ already occurs in Π. Closed processes correspond to the desired property of an extension E being closed under application of defaults in D.

Consider the default theory $T = (W, D)$ with $W = fag$ and D containing the following defaults:

$$\delta_1 = \frac{a : : b}{d}, \quad \delta_2 = \frac{true : c}{b}$$

$\Pi_1 = (\delta_1)$ is successful but not closed since δ_2 may be applied to $In(\Pi_1) = Th(fa, dg)$. $\Pi_2 = (\delta_1, \delta_2)$ is closed but not successful: both $In(\Pi_2) = Th(fa, d, bg)$ and $Out(\Pi_2) = fb, : cg$ contain b. On the other hand, $\Pi_3 = (\delta_2)$ is a closed and successful process of T. According to the following definition from Antoniou (1997), $In(\Pi_3) = Th(fa, bg)$ is an extension of T, in fact its single extension. Antoniou (1997) shows that this definition is equivalent to Reiter's original definition of extensions.

Definition 2.1 *A set of formulae E is an* extension of *the default theory T iff there is some closed and successful process Π of T such that E = In(Π).*

2.5 Some examples

Let $T = (W, D)$ with $W = ,$ and

$$D = \frac{^{n}true : a^{o}}{: a} :$$

T has no extensions. Indeed, the default may be applied because there is nothing preventing us from assuming a. But when the default is applied, the negation of a is added to the knowledge base, so the default invalidates its own application because both the *In*- and the *Out*-set contain $: a$. This example demonstrates that there need not always be an extension of a default theory.

Let $T = (W, D)$ be the default theory with $W = ,$ and $D = f\delta_1, \delta_2 g$ with

$$\delta_1 = \frac{true : p}{: q}, \quad \delta_2 = \frac{true : q}{r}$$

T has exactly one extension, namely $Th(f: qg)$, corresponding to the process (δ_1). Note that δ_1 can be applied after δ_2, but then $: q$ becomes part of the *In*-set, whilst it is also included in the *Out*-set (as the negation of the justification of δ_2). Thus (δ_2, δ_1) fails, and $Th(f: q, rg)$ is not an extension.

Let $T = (W, D)$ with $W = fgreen, aaaMemberg$ and $D = f\delta_1, \delta_2 g$ with

$$\delta_1 = \frac{green : : likesCars}{: likesCars}, \quad \delta_2 = \frac{aaaMember : likesCars}{likesCars}$$

T has exactly two extensions, one including *likesCars* and one including $: likesCars$. Note that the different extensions are due to the default rules supporting conflicting conclusions. This is not accidental. In fact, multiple extensions may only arise in the presence of contradictory defaults.

2.6 Properties of default logic

In this section we discuss some properties of the logic introduced above. The discussion in this section motivates alternative approaches that will be presented in subsequent sections. One point that should be stressed is that there is not a "correct" default logic approach, but rather the most appropriate for the concrete problem at hand. Different intuitions lead to different approaches that may work better for some applications and worse for others.

2.6.1 Existence of extensions

We saw that a default theory may not have any extensions. Is this a shortcoming of default logic? One might hold the view that if the default theory includes "nonsense" (for example $(true : p)/: p$), then the logic should indeed be allowed to provide no answer. According to this view, it is up to the user to provide meaningful information in the form of meaningful facts and defaults; after all, if a program contains an error, we don't blame the programming language.

The opposite view regards nonexistence of extensions as a drawback, and would prefer a more

"fault-tolerant" logic; one which works even if some pieces of information are deficient. This viewpoint is supported by the trend towards heterogeneous information sources, where it is not easy to identify which source is responsible for the deficiency, or where the single pieces of information are meaningful, but lead to problems when put together.

If we adopt the view that the possible nonexistence of extensions is a problem, then there are two alternative solutions. The first one consists in restricting attention to those classes of default theories for which the existence of extensions is guaranteed. Already in his classical paper, Reiter (1980) showed that if all defaults in a theory T are normal (in which case T is called a *normal default theory*), then T has at least one extension. Essentially this is because all processes are successful, as can be easily seen.

One problem with the restriction to normal default theories is that their expressiveness is limited. In general, it can be shown that normal default theories are strictly more expressive than general default theories. Normal defaults have limitations particularly regarding the interaction among defaults. Consider the example

Bill is a high school dropout.
Typically, high school dropouts are adults.
Typically, adults are employed.

These facts are naturally represented by the normal default theory T consisting of the fact $dropout(bill)$ and the defaults

$$\frac{dropout(X) : adult(X)}{adult(X)}, \frac{adult(X) : employed(X)}{employed(X)})$$

T has the single extension $Th(fdropout(bill), adult(bill), employed(bill)g)$. It is acceptable to assume that Bill is adult, but it is counterintuitive to assume that Bill is employed! That is, whereas the second default *on its own* is accurate, we want to prevent its application in case the adult X is a high school dropout. This can be achieved if we change the second default to

$$\frac{adult(X) : employed(X) \wedge : \ dropout(X)}{employed(X)}$$

But this default is not normal[2]. Defaults of this form are called *semi-normal*; Etherington (1987a) studied this class of default theories, and gave a sufficient condition for the existence of extensions. Another way of expressing interactions among defaults is the use of explicit priorities; this approach will be further discussed in section 3.

Instead of imposing restrictions on the form of defaults in order to guarantee the existence of extensions, we can also modify the concept of an extension in such a way that all default theories have at least an extension. In section 3 we will discuss two important variants with these properties, Lukaszewicz' Justified Default Logic and Schaub's Constrained Default Logic.

2.6.2 Joint consistency of justifications

It is easy to see that the default theory consisting of the defaults $(true : p)/q$ and $(true : : p)/r$ has the single extension $Th(fq, rg)$. This shows that the *joint consistency* of justifications is not required. Justifications are *not* supposed to form a set of beliefs, rather they are used to sanction "jumping" into some conclusions.

This design decision is natural and makes sense for many cases, but can also lead to unintuitive results. As an example consider the default theory, due to Poole, which says that, by default, a robot's arm (say a or b) is usable unless it is broken; further we know that either a or b is broken. Given this information, we would not expect both a and b to be usable.

[2] Note that it is unreasonable to add : $dropout(X)$ to the prerequisite of the default to keep it normal, because then we would have to definitely know that an adult is not a high school dropout before concluding that the person is employed.

Let us see how default logic treats this example. Consider the default theory $T = (W, D)$ with $W = \{broken(a) \vee broken(b)\}$ and D consisting of the defaults

$$\frac{true : usable(a) \wedge : \ broken(a)}{usable(a)}, \quad \frac{true : usable(b) \wedge : \ broken(b)}{usable(b)}$$

Since we do not have definite information that a is broken we may apply the first default and obtain $E' = Th(W \cup \{usable(a)\})$. Since E' does not include $broken(b)$ we may apply the second default and get $Th(W \cup \{usable(a), usable(b)\})$ as an extension of T. This result is undesirable, as we know that either a or b is broken.

In section 3 we shall discuss Constrained Default Logic (Schaub, 1992) as a prototypical default logic approach that enforces joint consistency of justifications of defaults involved in an extension.

The joint consistency property gives up part of the expressive power of default theories: under this property any default with several justifications $(\varphi : \psi_1, \ldots, \psi_n)/\chi$ is equivalent to the modified default $(\varphi : \psi_1 \wedge \ldots \wedge \psi_n)/\chi$ which has one justification. This is in contrast to a result in Besnard (1989) which shows that in Reiter's Default Logic, defaults with several justifications are strictly more expressive than defaults with just one justification. Essentially, in default logics adopting joint consistency it is impossible to express default rules of the form "In case I am ignorant about p (meaning that I know neither p nor $\neg p$) I conclude q". The natural representation in default form would be $(true : p, \neg p)/q$, but this default can never be applied if joint consistency is required, because its justifications contradict one another; on the other hand, it can be applicable in the sense of Reiter's Default Logic.

Another example for which joint consistency of justifications is undesirable is the following[3]. When I prepare for a trip then I use the following default rules:

If I may assume that the weather will be bad I'll take my sweater.
If I may assume that the weather will be good then I'll take my swimsuit.

In the absence of any reliable information about the weather I am cautious enough to take both with me. But note that I am not building a consistent belief set upon which I make these decisions; obviously the assumptions of the default rules contradict each other. So Reiter's default logic will treat this example in the intended way whereas joint consistency of justifications will prevent me from taking both my sweater and my swimsuit with me.

2.6.3 Cumulativity and lemmas

Cumulativity is, informally speaking, the property that allows for the safe use of lemmas. In mathematics, once a lemma has been proven it may be used in other proofs, without the effect that more or less conclusions can be drawn. Thus lemmas are used as shortcuts. In the following, we study an analogous property in the framework of default logic.

Let D be a fixed, countable set of defaults. For a formula φ and a set of formulae W we define $W \vdash_D \varphi$ iff φ is included in all extensions of the default theory (W, D). Now, cumulativity is the following property:

$$\textit{If } W \vdash_D \varphi, \ \textit{then for all } \psi: \ W \vdash_D \psi \Leftrightarrow W \cup \{\varphi\} \vdash_D \psi$$

If we interpret φ as a lemma, cumulativity says that the same formulae can be obtained from W as from $W \cup \{\varphi\}$. This is the standard basis of using lemmas in, say, mathematics. Default logic does not respect cumulativity: consider $T = (W, D)$ with $W = \emptyset$ and D consisting of the defaults

$$\frac{true : a}{a}, \quad \frac{a \wedge b : : \ a}{: \ a}$$

(this example is due to Makinson). The only extension of T is $Th(\{a\})$. Obviously, $W \vdash_D a$. From $a \wedge b \in Th(\{a\})$ we get $W \vdash_D a \wedge b$. If we take $W' = \{a \wedge b\}$, then the default theory (W', D) has two extensions, $Th(\{a\})$ and $Th(\{\neg : a, b\})$; therefore, $W \cup \{a \wedge b\} \not\vdash_D a$.

[3] My thanks go to an anonymous referee.

Quite some work has been invested in developing default logics that possess the cumulativity property, one notable approach being Brewka's Cumulative Default Logic (Brewka, 1991). But it is doubtful whether this is the right way to go.

An alternative approach was taken by Schaub (1992) who proposed to represent a lemma by a corresponding *lemma default* which records in its justifications the assumptions on which a conclusion was based. The formal definition of a lemma default is as follows.

Let Π_χ be a nonempty, minimal successful process of T such that $\chi \in In(\Pi_\chi)$. A *lemma default* δ_χ *corresponding to* χ is the default

$$\frac{true : \psi_1, \ldots, \psi_n}{\chi}$$

where $\{\psi_1, \ldots, \psi_n\} = \{\psi \mid \psi \in just(\delta)\}$ for a δ occurring in Π_χ. This default collects all assumptions that were used in order to derive χ.

Theorem 2.2 *Let χ be included in an extension of T and δ_χ a corresponding lemma default. Then T and $T' = (W, D \cup \{\delta_\chi\})$ have the same extensions.*

The proof can be found in Schaub (1992, 1997) and Antoniou (1997). So it is indeed possible to represent lemmas in default logic, not as facts (as required by cumulativity) but rather as defaults, which appears more natural anyway, since it highlights the nature of a lemma as having been proven defeasibly and thus as being open to disputation. Of course, cumulativity is a desirable and elegant property from an abstract view of nonmonotonicity (Makinson, 1994).

2.7 Sceptical versus credulous reasoning

So far we have described the meaning of a default theory using extensions. These are valuable because they give an overview of the possible interpretations of the given knowledge. For example, in diagnostic problems an extension represents a possible diagnosis of the system. In requirements engineering an extension may represent a maximal set of requirements that can be simultaneously satisfied (see also section 6.1). But in some cases, we may be interested in which *conclusions* are supported by the given knowledge. In principle, there are two extreme approaches.

In *sceptical reasoning* we accept only those conclusions which are true in all extensions of the default theory. It is a cautious approach, which takes the view that since we don't know which is the right extension, we should only accept their intersection. In many problems this seems the appropriate way, for example when definitive advice is required (e.g., advice on the applicability of regulations).

On the other hand, in the *credulous approach* we accept conclusions which are included in at least one extension. According to this view, it is sufficient to have an argument supporting the conclusion. This approach is appropriate in problems such as the preparation of a court case, where one is interested in establishing that a claim can be substantiated. Which of the two approaches is the right one depends on the specific problem at hand.

3 Variants of default logic

We briefly review some basic default logic variants. Other well-known variants include Cumulative Default Logic (Brewka, 1991), Disjunctive Default Logic (Gelfond et al., 1991) and weak extensions (Marek & Truszczynski, 1993).

3.1 Justified default logic

Lukaszewicz considered the possible nonexistence of extensions as a representational shortcoming of the original default logic, and presented a variant, Justified Default Logic (Lukaszewicz, 1988) which avoids this problem. The essence of his approach is the following: If we have a successful but not yet closed process, and all ways of expanding it by applying a new default leads to a failed

process, then we stop and accept the current In-set as an extension. In other words, we take back the final, "fatal" step that causes failure. Consider the default theory $T = (W, D)$ with $W = fholidays, sundayg$ and D consisting of the defaults

$$\delta_1 = \frac{sunday : goFishing \wedge : \ wakeUpLate}{goFishing}, \quad \delta_2 = \frac{holidays : wakeUpLate}{wakeUpLate}$$

It is easily seen that T has only one extension (in the sense of section 2), namely $Th(fholidays, sunday, wakeUpLateg)$. But if we apply δ_1 first, then δ_2 can be applied and leads to a failed process. In this sense we lose the intermediate information $Th(fholidays, sunday, goFishingg)$. On the other hand, in Justified Default Logic we would stop after the application of δ_1 instead of applying δ_2 and running into failure; therefore $Th(fholidays, sunday, goFishingg)$ is an additional (modified) extension.

Technically, this is achieved by paying attention to maximally successful processes. Let T be a default theory, and let Π and Γ be processes of T. We define $\Pi < \Gamma$ iff the set of defaults occurring in Π is a proper subset of the defaults occurring in Γ. Π is called a *maximal process of* T, iff Π is successful and there is no successful process Γ such that $\Pi < \Gamma$. A set of formulae E is called a *modified extension* of T iff there is a maximal process Π of T such that $E = In(\Pi)$.

In the example above, $\Pi = (\delta_1)$ is a maximal process: the only process that strictly includes Π is $\Gamma = (\delta_1, \delta_2)$ which is not successful. Therefore $Th(fholidays, sunday, goFishingg)$ is a modified extension of T. T has another modified extension, which is the single extension $Th(fholidays, sunday, wakeUpLateg)$ of T. Obviously, every closed and successful process is a maximal process (since no new default can be applied). Also, any process can be successfully extended to a maximal process. Therefore, we have the following result (Lukaszewicz, 1988):

Theorem 3.1 *Let T be a default theory. Then T has at least one modified extension. Also, every extension of a default theory T is a modified extension of T.*

3.2 Constrained default logic

Justified Default Logic avoids running into inconsistencies, and can therefore guarantee the existence of modified extensions. On the other hand, it does not require joint consistency of default justifications; for example, the default theory $T = (W, D)$ with $W =$, and $D = ftrue : p/q, true : : p/rg$ has the single modified extension $Th(fq, rg)$. Constrained Default Logic (Schaub, 1992; Delgrande et al., 1994) is a default logic approach which enforces joint consistency. In the example above, after the application of the first default the second default may not be applied because p contradicts $: p$.

Furthermore, since the justifications are consistent with each other, we test the consistency of their conjunction with the current knowledge base. In the terminology of processes, we require the consistency of $In(\Pi) \flat : Out(\Pi)$[4].

Finally, we adopt the idea from the previous section, namely a default may only be applied if it does not lead to a contradiction (failure) a posteriori. That means, if $(\varphi : \psi_1, \triangleright\triangleright\triangleright, \psi_n)/\chi$ is tested for application to a process Π, then $In(\Pi) \flat : Out(\Pi) \flat f\psi_1, \triangleright\triangleright\triangleright, \psi_n, \chi g$ must be consistent. We note that the set Out no longer makes sense since we require joint consistency. Instead, we have to maintain the set of formulae which consists of W, all consequents *and all justifications* of the defaults that have been applied.

- Given a default theory $T = (W, D)$ and a sequence Π of defaults in D without multiple occurrences, we define $Con(\Pi) = Th(W \flat f\varphi \ j \ \varphi$ is the consequent or a justification of a default occurring in $\Pi g)$. Sometimes we refer to $Con(\Pi)$ as the *set of constraints* or the *set of supporting beliefs*.

[4] : $Out(\Pi)$ is the set $f: \ \varphi \ j \ \varphi \ 2 \ Out(\Pi)g$.

$Con(\Pi)$ represents the set of beliefs supporting Π. Thus a *constrained extension* is a pair (E, Con). The first component is the knowledge offered by the extension, while Con is the consistent set of supporting assumptions.

Let us reconsider the "broken arms" example: $T = (W, D)$ with $W = \{broken(a) \wedge broken(b)\}$, and D consisting of the defaults

$$\delta_1 = \frac{true : usable(a) \wedge : \ broken(a)}{usable(a)}, \quad \delta_2 = \frac{true : usable(b) \wedge : \ broken(b)}{usable(b)}$$

It is easily seen that there are two closed constrained processes, (δ_1) and (δ_2), leading to two constrained extensions:

$(Th(W \cup \{usable(a)\}), Th(\{broken(b), usable(a), : \ broken(a)\}))$, and
$(Th(W \cup \{usable(b)\}), Th(\{broken(a), usable(b), : \ broken(b)\}))$.

The effect of the definitions above is that it is impossible to apply both defaults together: after the application of, say, δ_1, $: \ broken(a)$ is included in the Con-set; together with $broken(a) \wedge broken(b)$ it follows $broken(b)$, therefore δ_2 is blocked. The two alternative constrained extensions describe the two possible cases we would have intuitively expected.

For another example consider $T = (W, D)$ with $W = \{p\}$ and

$$D = \{\frac{p : : r}{q}, \frac{p : r}{r} : \}^{\rho, \sigma}$$

T has two constrained extensions, $(Th(\{p, q\}), Th(\{p, q, : \ r\}))$ and $(Th(\{p, r\}), Th(\{p, r\}))$. Note that for both constrained extensions, the second component collects the assumptions supporting the first component.

Theorem 3.2 *Every default theory has at least one constrained extension. Furthermore, Constrained Default Logic is semi-monotonic.*

The proof is found in Delgrande et al. (1994). In the following, we collect some interconnections among the default logic variants presented so far. Their proofs are found in Antoniou (1997).

Theorem 3.3 *Let T be a default theory and $E = In(\Pi)$ an extension of T, where Π is a closed and successful process of T. If $E \cup \{: \ Out(\Pi)\}$ is consistent, then $(E, Th(E \cup \{: \ Out(\Pi)\}))$ is a constrained extension of T.*

The converse does not hold, since the existence of an extension is not guaranteed. For example $T = (\{\}, \{f(true : p)/: \ p\})$ has the single constrained extension $(Th(\{\}), Th(\{\}))$, but no extension.

Theorem 3.4 *Let T be a default theory and $E = In(\Pi)$ a modified extension of T, where Π is a maximal process of T. If $E \cup \{: \ Out(\Pi)\}$ is consistent, then $(E, Th(E \cup \{: \ Out(\Pi)\}))$ is a constrained extension of T.*

The example $T = (W, D)$ with $W = \{\}$ and $D = \{f(true : p)/q, (true : : p)/r\}$ shows that we cannot expect the first component of a constrained extension to be a modified extension: T has the single modified extension $Th(\{q, r\})$, but possesses two constrained extensions, $(Th(\{q\}), Th(\{p, q\}))$ and $(Th(\{r\}), Th(\{f : p, r\}))$. As the following result demonstrates, it is not accidental that for both constrained extensions, the first component is included in the modified extension.

Theorem 3.5 *Let T be a default theory and (E, C) a constrained extension of T. Then there is a modified extension F of T such that $E \subseteq F$.*

The following examples illustrates well the difference between the three approaches. Consider the default theory $T = (W, D)$ with $W = \{\}$ and

$$D = \{\frac{true : p}{q}, \frac{true : : p}{r}, \frac{true : : q, : r}{s}\}^{\rho, \sigma}$$

T has the single extension

$Th(fq, rg)$
two modified extensions,
$Th(fq, rg)$
$Th(fsg)$
and three constrained extensions
$(Th(fqg), Th(fq, pg))$
$(Th(frg), Th(fr, : pg))$
$(Th(fsg), Th(fs, : q, : rg))$
This theory illustrates the essential differences of the three approaches discussed. Default logic does not care about inconsistencies among justifications and may run into inconsistencies. Thus the first two defaults can be applied together, while if the third default is applied first, then the process is not closed and subsequent application of another default leads to failure. Justified Default Logic avoids the latter situation, so we obtain an additional modified extension. Constrained Default Logic avoids running into failure, too, but additionally requires joint consistency of justifications, therefore the two first defaults cannot be applied in conjunction, as in the other two approaches. Thus, we get three constrained extensions.

We conclude this section by noting that for normal default theories all default logic approaches discussed are identical. In other words, they coincide for the "well-behaved" class of default theories, and seek to extend it in different directions.

Theorem 3.6 *Let T be a normal default theory, and E a set of formulae. The following statements are equivalent:*

(a) *E is an extension of T.*
(b) *E is a modified extension of T.*
(c) *There exists a set of formulae C such that (E, C) is a constrained extension of T.*

3.3 Rational default logic

Constrained Default Logic enforces joint consistency of the justifications of defaults that contribute to an extension, but goes one step further by requiring that the consequent of a default be consistent with the current *Con*-set. Rational Default Logic (Mikitiuk & Truszczynski, 1995) does not require the latter step. Technically, a default $(\varphi : \psi_1, \triangleright\triangleright\triangleright, \psi_n)/\chi$ is *rationally applicable* to a pair of deductively closed sets of formulae (E, C) iff $\varphi \supseteq E$ and $f\psi_1, \triangleright\triangleright\triangleright, \psi_n g\flat\ C$ is consistent.

As an example, consider the default theory $T = (W, D)$ with $W =$, and

$$D = \overset{\rho}{\frac{true : b}{c}}, \frac{true : : b}{d}, \frac{true : : c}{e}, \frac{true : : d}{f}\overset{\sigma}{} :$$

T has the single extension $Th(fc, dg)$, three constrained extensions,

$(Th(fe, fg), Th(fe, : c, f, : dg))$
$(Th(fc, fg), Th(fc, b, f, : dg))$
$(Th(fd, eg), Th(fd, : b, e, : cg))$

but two rational extensions, $Th(fc, fg)$ and $Th(fd, eg)$. The first constrained extension is "lost" in Rational Default Logic because it is not closed under application of further defaults. $(true : b)/c$ and $(true : : b)/d$ are both rationally applicable to $Th(fe, fg)$; but once one of them is applied to $Th(fe, fg)$ we get a failed situation.

Mikitiuk & Truszczynski (1995) shows that if E is an extension of T in Rational Default Logic, then (E, C) is a constrained extension of T for some set C. The converse is true for semi-normal default theories.

Rational Default Logic does not guarantee the existence of extensions. For example, the default theory consisting of the single default $(true : p)/: p$ does not have any extensions.

3.4 Priorities

When pieces of information are not certain, as is the case with defaults, it is natural to rank them according to the degree of belief in them. Technically, this is done by imposing a preference (priority) relation $<$ on defaults. Priorities arise naturally in many application domains. They can be based on specificity, authority (e.g., Federal law overrides State law), freshness (a more recent law overrides an older law), or reliability of the information sources. For example, suppose we have the information

$$bird$$
$$penguin$$
$$\delta_1 = \frac{bird : flies}{flies}$$
$$\delta_2 = \frac{penguin : : flies}{: flies}$$
$$\delta_1 < \delta_2$$

Then there is only one (preferred) extension, namely the one which includes $: flies$. This is so because δ_2 is preferred over δ_1. This preference is based on specificity: While δ_1 is a general rule concerning all birds, δ_2 is more specific, in that it applies to penguins only; thus, it seems natural to assume that it is more relevant to the case of penguins.

The main drawback of the idea of having static priority information is that the user has to provide all priority information, which is supposed to be static and not change over time. These requirements may be too strict for some problems. Brewka (1994) proposed an approach in which priority information is part of the logical language and can be derived as any other statements. For example, it is possible to write

$$\frac{p : d_1 < d_2}{d_1 < d_2}$$

to express that "In cases p is true, usually default d_2 should be given preference over default d_1". Of course, there may be some exceptions in which $d_1 < d_2$ cannot be assumed due to other information being available.

As this example shows, the approach uses *names* to refer to defaults, and a preference relation $<$ as part of the language. Reasoning about priorities is very flexible, but the high expressiveness brings problems, too; for example, it is possible that even if all defaults are normal there is no extension. Consider the theory consisting of the following two defaults:

$$d_1\ \eta\ \frac{true : d_1 < d_2}{d_1 < d_2}, \quad d_2\ \eta\ \frac{true : d_2 < d_1}{d_2 < d_1}$$

Default d_1 says that usually you should prefer default d_2 instead of d_1, and *vice versa*. Therefore, this theory has no extension.

4 Semantic characterisations

4.1 Argumentation

Argumentation provides an abstract view of nonmonotonic reasoning. It is based on the consideration of arguments and their possible defeat by counterarguments. In the following, we briefly describe the characterisation of default logic in the argumentation framework of Bondarenko et al. (1997).

Underlying any argumentation framework is a *deductive basis*, which consists of the logical language L, and a set R of inference rules. The deductive basis defines a syntactic provability relation ℓ; in this section $Th(T)$ denotes the set $f\alpha\ 2\ L\ j\ T\ \ell\ \alpha g$.

An *Argumentation-Based Framework* (w.r.t. a deductive basis) consists of

- a set W of formulae, representing the certain knowledge.
- a set Ab of formulae, representing the possible assumptions.
- a function $^{\Gamma} : Ab\ \omega\ \ For$, with the idea that $\overline{\alpha}$ represents the contrary of $\alpha\ 2\ Ab$.

A set $\Delta \subseteq Ab$ *attacks* an assumption $\alpha\ 2\ Ab$ iff $W \flat\ \Delta\ \ell\ \overline{\alpha}$. A set of assumptions Δ attacks a set of assumptions Δ' iff there is an $\alpha\ 2\ \Delta'$ such that Δ attacks α. A set of assumptions $\Delta \subseteq Ab$ is *stable* iff

(a) Δ is *closed*: $\Delta = f\alpha\ 2\ Ab\ j\ T \flat\ \Delta\ \ell\ \alpha g$.
(b) Δ does not attack itself.
(c) Δ attacks every α 6 Δ.

If Δ is stable, then $Th(W \flat\ \Delta)$ is called a *stable extension*.

The concepts of attack and stability are independent of the particular argumentation-based framework. In fact they have been used to characterise several nonmonotonic reasoning approaches; see Bondarenko et al. (1997) for details. In the following, we show how default logic can be embedded into this framework.

Let $T = (W, D)$ be a default theory. We define its translation $arg(T)$ into the argumentation-based framework. First we define the deductive basis. The language L consists of the first order language L_0 of T, extended by additional predicate symbols $M\alpha$ for every closed formula α in the language of T. Let R_0 be a deductively complete set of inference rules for predicate logic (in the language L_0). We add the following inference rules, which correspond to each of the defaults in T. Essentially, we wish to infer the consequent of a default if we have assumed all its justifications and we have already inferred its prerequisite. Formally[5]:

$$L = L_{\prime} \flat\ fM\alpha\ j\ \alpha \text{ is closed formula in } L_{\prime} g.$$

$$R = R_0 \flat\ \overset{\rho}{\frac{\varphi, M\psi_1, \triangleright\triangleright\triangleright, M\psi_n}{\chi}}\ j\ \overset{\sigma}{\frac{\varphi : \psi_1, \triangleright\triangleright\triangleright, \psi_n}{\chi}}\ 2\ D$$

The argumentation-based framework $arg(T) = (W, Ab, ^{\Gamma})$ is now defined as follows:

- $Ab = fM\psi\ j\ \psi\ 2\ just(\delta)$ for $\delta\ 2\ Dg$.
- $\overline{M\alpha} =: \alpha$.

Theorem 4.1 *E is an extension of T iff there is a stable extension E' of $arg(T)$ such that $E = E' \natural\ L_{\prime}$.*

The proof is found in Bondarenko et al. (1997). As an example, consider the default theory with the two defaults

$$\frac{true : p}{q}, \quad \frac{true : r}{: p}$$

In the translation we have

$$Ab = fMp, Mrg$$

$$R = R_{\prime} \flat\ \overset{\rho}{\frac{Mp}{q}}, \overset{\sigma}{\frac{Mr}{: p}}$$

$\Delta = fMrg$ is stable: (i) it is closed since we can only infer the assumption Mr using Δ (this is true for any default theory); (ii) it does not attack itself; (iii) it attacks the assumption Mp not in Δ: $\Delta\ \ell\ : p = \overline{Mp}$.

On the other hand, $\Delta' = fMp, Mrg$ is not stable because it attacks itself: $\Delta'\ \ell\ : p = \overline{Mp}$ and $Mp\ 2\ \Delta'$. Finally, $\Delta'' = fMpg$ is not stable because it does not attack Mr which is not included in Δ''.

[5] Note that the rules in R are not defaults, but rather inference rules in the sense of classical logic: if all formulas above the line have been derived, then we may also derive the formula below the line.

4.2 Operational semantics

In this section we will give a semantic characterisation of default logic in the semantic framework of Teng (1996). Consider the default theory T consisting of the fact p and the defaults $\delta_1 = (p : q)/q$ and $\delta_2 = (q : r)/r$. Obviously T has the single extension $Th(fp, q, rg)$. It is obtained from the process (δ_1, δ_2).

Let W be the set of all possible (total) worlds over the propositional language fp, q, rg. The semantic counterpart of the process above is the following so-called *default partition sequence*:

$S =< W_{\prime}, W_1, W_2, W_3 >$, where
$W_0 = fw\ j\ w \models : pg$
$W_1 = fw\ j\ w \models (p \wedge : q)g$
$W_2 = fw\ j\ w \models (p \wedge q \wedge : r)g$
$W_3 = fw\ j\ w \models (p \wedge q \wedge r)g$

First note that S forms a partition of W. $W_{\prime}$ includes all worlds in which the fact p is false. W_1 includes those worlds not in $W_{\prime}$ in which the consequent of the first default is false. Equally, W_2 contains those worlds not in $W_{\prime} \flat W_1$ in which the consequent of the second default is not true. Finally, W_3 consists of the remaining worlds, in which the fact and the consequents of the defaults applied are true. The set of all formulae true in W_3 is an extension of T, indeed its only extension.

Essentially $< W_i, \bowtie, W_l >$ defines a "frame of reference", that is, the body of knowledge which builds the current context before applying the ith rule. Each reasoning step is performed with respect to the current context. Once an inference is made, the frame of reference is updated by pruning out additional worlds, those in which the new conclusion is false.

Note also that the prerequisite p of the first default applied is true in all worlds in $W_1 \flat W_2 \flat W_3$, and that there is at least one world in W_3 in which the justification of δ_1 is true. This reflects the property of success in the process model of default logic: when a default is applied, its justifications must be consistent not only with the current knowledge base, but also with the final result of the branch of the process tree (that is, with the *In*-set of the closed process).

For simplicity, we give the semantics in the propositional case. Let Σ be a propositional signature, that is, a set of propositional atoms. We call w a *world* (in Σ) iff $w \subseteq \Sigma \flat : \Sigma$, and for every $p \in \Sigma$ either $p \in w$ or $: p \in w$. $w \models \varphi$ denotes validity of the formula φ in w. Let W be the set of all possible worlds (in Σ).

A *partition sequence* of W is a tuple $< W_1, \bowtie, W_l >$ $(l \leq 1)$ such that the non-empty elements W_i form a partition of W.

Definition Let $T = (W, D)$ be a default theory theory. A *default partition sequence* for T is a partition sequence $S =< W_{\prime}, \bowtie, W_l >$ such that there is a sequence of $P =< \delta_1, \bowtie, \delta_{l\Gamma_1} >$ satisfying the following conditions:

(a) For all $i = 1, \bowtie, l\ \Gamma\ 1$, $W_i = fw\ j\ w \in W_{\prime} \flat \bowtie\flat W_{i\Gamma_1}$ and $w \not\models cons(\delta_i)g$
(b) For all $i = 1, \bowtie, l\ \Gamma\ 1$:

  (i) $\forall w \in W_i \flat \bowtie\flat W_l : w \models pre(\delta_i)$
  (ii) $\forall \psi \in just(\delta_i) \exists w \in W_l : w \models \psi$

(c) There is no default $\delta \notin f\delta_1, \bowtie, \delta_{l\Gamma_1}g$ which is applicable in the sense of (b) (replacing δ_i by δ, and i by l).

Condition (a) ensures that every time a default is applied, the worlds in which its consequent is false are disregarded from further consideration. Condition (b) ensures applicability of the respective default in the current frame of reference. Finally (c) corresponds to the closure property of processes.

Theorem 4.2 *If E is an extension of T, then there is a default partition sequence $S =< W_{\prime}, \bowtie, W_l >$ of T such that $E = f\varphi\ j\ \forall w \in W_l : w \models \varphi g$.*

5 Logic programming-based default reasoning

5.1 Stable models of logic programs

Sometimes it is convenient to express information in the form of logic programs. Moreover, logic programming offers a computational basis for logical reasoning in general, and default reasoning in particular. The treatment of negation in logic programming is closely related to nonmonotonicity – in fact, *negation as failure* was an early formalism with nonmonotonic flavour.

5.1.1 Basics of logic programming

A *logic program clause* is an expression of the form

$$A \psi \quad B_1, \triangleright\triangleright\triangleright, B_n, not\ C_1, \triangleright\triangleright\triangleright, not\ C_m$$

where $A, B_1, \triangleright\triangleright\triangleright, B_n, C_1, \triangleright\triangleright\triangleright, C_m$ are atomic formulae, and $n, m \lambda\ 0$. A is called the *head* of the clause, and $B_1, \triangleright\triangleright\triangleright, B_n, not\ C_1, \triangleright\triangleright\triangleright, not\ C_m$ form the *body*. If all formulae involved in the clause are ground then the clause is called *ground*. If $m = 0$ then the clause is called *definite*. If $n = m = 0$, then the clause is called a *fact*, otherwise a *rule*.

A *logic program P* is a set of program clauses. A *definite logic program* (or *Horn logic program*) is a logic program which consists of definite program clauses only.

Now we give an interpretation of logic programs based on predicate logic; it is determined by considering the following representation of a program clause in first-order logic:

$$8(A \psi \quad B_1 \wedge \triangleright\triangleright\triangleright \wedge B_n \wedge : C_1 \wedge \triangleright\triangleright\triangleright \wedge : C_m):$$

That is, a clause is implicitly assumed to be universally quantified, with quantifiers outmost. The *predicate logic interpretation* of a logic program P is denoted by $pc(P)$, and is the first-order theory which consists of the predicate logic interpretation of all program clauses in P.

Let P be a logic program. Every model of the first-order theory $pc(P)$ is called a *model of P*. Every Herbrand model of $pc(P)$ is called a *Herbrand model of P*. A definite logic program P has a unique least Herbrand model, denoted by M_P.

In the definition of stable models to follow, we make use of the set of ground instances of a logic program P, resulting in a ground logic program which is denoted by $ground(P)$. It can be shown that for a logic program P, the (predicate logic) models of $pc(ground(P))$ are exactly the Herbrand models of P.

5.1.2 Stable models of logic programs

Given a program clause

$$A \psi \quad B_1, \triangleright\triangleright\triangleright, B_n, not\ C_1, \triangleright\triangleright\triangleright, not\ C_m$$

the predicate logic interpretation treats the negation symbol *not* as classical negation: in order to conclude A, we have to know (or prove) that $: C_i$ is true, for all i 2 $f1, \triangleright\triangleright\triangleright, mg$.

What is "wrong" with this interpretation? Well, in many cases we may wish to interpret *not* in a different way. For example, we may wish to interpret (or assume) *not connection*(*newcastle*, *sydney*, 12*pm*) to be "true" if we look up the timetable, and do not find this train connection (as opposed to actually *knowing* that there is no such connection). This type of negation is called *negation as failure*, and its nonmonotonic character is obvious: a change in the timetable may invalidate a previous conclusion which was based on the absence of some information.

This idea resembles the intuitive interpretation of a default in default logic. The stable semantics uses this observation to assign meaning to logic programs. In particular, it uses the following translation of logic programs into default theories.

The Default Logic interpretation of a ground program clause *Cl*

$$A \psi \quad B_1, \triangleright\triangleright\triangleright, B_n, not\ C_1, \triangleright\triangleright\triangleright, not\ C_m$$

is given by the default[6]

$$df(Cl) = \frac{B_1 \wedge \triangleright\triangleright\triangleright \wedge B_n : : \ C_1, \triangleright\triangleright\triangleright, : \ C_m}{A}$$

We define $df(P)$, the *default logic interpretation* of the logic program P, to be the default theory (W, D) with $W = ,$ and $D = fdf(Cl) \ j \ Cl \ 2 \ ground(P)g.$

Definition Let M be a subset of the Herbrand base. M is a *stable model* of the logic program P iff $Th(M)$ is an extension of $df(P)$.

Theorem 5.1 *Let P be a logic program. A stable model of P is a minimal Herbrand model of P. In particular, if P is a definite logic program, then M P is the unique stable model of P.*

The proof is found in Marek & Truszczynski (1993). The converse of this result is not true, which means that not every minimal Herbrand model of P is a stable model of P. To see this, consider the logic program consisting of the single clause $p \ \psi \ \ not \ p$. P has the minimal Herbrand model fpg, but the corresponding default theory

$$\subseteq \ \ \rho \ , \ , \ \frac{true : : \ p}{p}^{\sigma\iota}$$

has no extension, so P has no stable model.

5.2 Defeasible logic

Defeasible logic (Nute, 1987, 1994) is an approach to default reasoning which is based on logic programming and came with an implementation right from the beginning. Its basic idea is to use simple rules *without negation as failure*, plus a superiority relation among rules. Defeasible logic is a simple knowledge representation formalism. It avoids many features which make other default reasoning approaches computationally and conceptually complex. Defeasible logic is not based on extensions, but supports only *sceptical reasoning*.

We begin by presenting its basic ingredients. A defeasible theory (a knowledge base in Defeasible Logic) consists of five different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, and a superiority relation.

Strict rules are rules in the classical sense: whenever the premises of a rule are given, we are allowed to apply the rule and get a conclusion. When the premises are indisputable (e.g., facts) then so is the conclusion. An example of a strict rule is "Emus are birds". Written formally,

$$emu(X) \ \omega \ \ bird(X)$$

Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is "Birds typically fly"; written formally:

$$bird(X) \leftarrow flies(X)$$

The idea is that if we know that something is a bird, then we may conclude that it flies, *unless there is other evidence suggesting that it may not fly*.

Defeaters are rules that cannot be used to draw any conclusions. Their only use is to prevent some conclusions. In other words, they are used to defeat some defeasible rules by producing evidence to the contrary. An example is "If an animal is heavy then it may not be able to fly". Formally:

$$heavy(X) \rightsquigarrow : \ flies(X)$$

The main point is that the information that an animal is heavy is not sufficient evidence to conclude

[6] In case $n = 0$ or $m = 0$, the prerequisite resp. justification of $df(Cl)$ is the formula *true*.

that it doesn't fly. It is only evidence that the animal *may* not be able to fly. In other words, we don't wish to conclude : $flies(X)$ in this case.

The *superiority relation* among rules is used to define priorities among rules, that is, where one rule may override the conclusion of another rule. For example, given the defeasible rules

$$\begin{array}{lll} r: & bird(X) & \leftarrow flies(X) \\ r': & brokenWing(X) & \leftarrow : flies(X) \end{array}$$

which contradict one another, no conclusive decision can be made about whether a bird with broken wings can fly. But if we introduce a superiority relation $>$ with $r' > r$, then we can indeed conclude that it can't fly.

It turns out that we only need to define the superiority relation over rules with contradictory conclusions. Also notice that a cycle in the superiority relation is counterintuitive from the knowledge representation perspective. In the above example, it makes no sense to have both $r > r'$ and $r' > r$. Consequently, the defeasible logic we discuss requires an acyclic superiority relation.

Example

$quaker(nixon)$
$republican(nixon)$
$r_1: quaker(X) \leftarrow : militarist(X)$[7]
$r_2: republican(X) \leftarrow militarist(X)$

If the superiority relation is empty, then we cannot prove either $militarist(nixon)$ nor $: militarist(nixon)$ (since there is no reason for preferring one rule over the other – this is a manifestation of sceptical reasoning). But if we add $r_2 > r_1$, then we can prove $militarist(nixon)$.

Example We borrow this example from Grosof (1996).

$quaker(nixon)$
$republican(nixon)$
$r_1: quaker(X) \leftarrow pacifist(X)$
$r_2: republican(X) \leftarrow : pacifist(X)$
$r_3: republican(X) \leftarrow footballfan(X)$
$r_4: pacifist(X) \leftarrow antimilitary(X)$
$r_5: footballfan(X) \leftarrow : antimilitary(X)$
The superiority relation is empty.

Here we can prove : $antimilitary(nixon)$. This occurs by proving $footballfan(nixon)$, and showing that $pacifist(nixon)$ is not provable (the rules r_1 and r_2 contradict one another, and there is no priority information to select one of them; since defeasible logic is sceptical, this means that neither $pacifist(nixon)$ nor : $pacifist(nixon)$ can be proven). The latter makes sure that r_4 cannot be applied, thus there is no conflict with rule r_5, which therefore fires. This behaviour should be contrasted with approaches based on extensions, e.g. default logic. The following default theory corresponds to the defeasible knowledge base above.

$$\frac{quaker(X) : pacifist(X)}{pacifist(X)}, \frac{republican(X) : : pacifist(X)}{: pacifist(X)}, \frac{republican(X) : footballfan(X)}{footballfan(X)},$$

$$\frac{pacifist(X) : antimilitary(X)}{antimilitary(X)}, \frac{footballfan(X) : : antimilitary(X)}{: antimilitary(X)}$$

There are three extensions:

One including $pacifist(nixon)$, $antimilitary(nixon)$

[7] Rules with free variables are interpreted as rule schemas, that is, as the set of all ground instances.

One including *pacifist*(*nixon*), *footballfan*(*nixon*), : *antimilitary*(*nixon*)
One including : *pacifist*(*nixon*), : *antimilitary*(*nixon*)

Note that : *antimilitary*(*nixon*) is *not* included in the intersection of these extensions. The difference between these two kinds of sceptical reasoning has already been pointed out in Touretzky et al. (1987).

6 Applications

Default reasoning has great potential to be used in practical applications. It is a natural way of thinking about problems, not too complex (for example, no consideration of probabilities or fuzziness measures are necessary), and default rules prevail in many application domains. The reason that default reasoning has not taken off yet is the lack of implementations and usable tools. But with the emergence of powerful systems (see conclusions section) these tools will soon be available.

Instead of trying to give an overview of actual or potential applications, in this section we will discuss three application areas which we believe are typical of default reasoning applications of the future: requirements engineering, legal reasoning, and the control of technical systems.

6.1 Requirements engineering

Software engineering is concerned with the specification, development, validation and maintenance of software systems. The construction typically commences with a collection of activities called *requirements engineering*. The major objective of requirements engineering is to define the purpose and external behaviour of the software system.

It is desirable that the product of requirements engineering should be a *formal specification* capturing customer requirements. The formality of such a specification makes it amenable to rigorous analysis techniques which, in turn, facilitate the *validation* of customer requirements. Furthermore, the formality of specifications makes them better usable at later stages of the software development process, and supports the use of formal methods in the development and verification of the software system. The process of translating informal, vague, possibly conflicting requirements into a formal specification is a demanding task. Here are some central issues that arise:

1 *Inconsistency management*. During requirements elicitation it is almost inevitable that inconsistencies arise. They may be the result of mistakes, misunderstandings, or lack of information, especially where a specification is developed collaboratively. They may be the result of infeasible or impractical requirements, or due to conflicting information sources, or simply because different potential users of a software system have different views and interests.

   Issues of inconsistency handling have been subject to thorough investigation. For example, the ViewPoints methodology Nuseibeh (1996), provides a framework for requirements elicitation, by allowing to express the relationships between multiple views, while Balzer (1991), Boehm & In (1996) and Hunter & Nuseibeh (1997) deal with inconsistency handling.
2 *Evolving requirements and specifications*. As Zowghi & Offen (1997) point out, software development is characterized by continuous evolution. Requirements evolve because requirements engineers and users cannot possibly foresee all the ways in which a software system can be utilised. It is very likely that the software will eventually be used by different people with differing goals and needs than those identified during the initial planning of the software. And, of course, changes in requirements cause changes in design which lead to changes in the implemented system. Thus support for the evolution of software is clearly needed, and this is particularly true for the requirements phase, since changes in requirements typically initiate change throughout the software development life cycle.
3 *Quality of specifications*. Properties of specifications that are important include the naturalness of expression, understandability, ease of maintenance, and compactness.

Default logic addresses the main problems outlined in the previous subsection. The inconsistency problem is resolved by the possibility of a specification to have several extensions. In default logic, there is no trivialisation effect in the face of an inconsistency, if it is caused by the use of default rules. Instead, there are more than one extensions, each of them representing an alternative, consistent interpretation. In this sense, the set of extensions provides a good overview of the conflicts and possible ways of resolving them.

Regarding the evolution problem, the development of requirements, specifications and software is by itself a nonmonotonic process, in which some initial project steps may have to be taken back. Thus a nonmonotonic formalism, such as default logic, has some advantages to represent these activities. The evolution of specifications is a revision process, which is subject to rigorous information change techniques (Gardenfors, 1988); these techniques were shown to be closely related to default reasoning (Gardenfors & Makinson, 1994). Also, (Zowghi et al. (1996) describes information change techniques for specifications with default rules.

6.2 Simple specification of controllers using default rules

Covington (1997) describes an interesting application of the use of default rules in an embedded control system. A prototype air conditioner controller was designed using defeasible logic, and was implemented on a PIC16F84 microcontroller. The PIC16F84 has 1K words of program memory, 64 bytes of RAM, and runs at 0.1 MHz. Currently, it costs around $6.5. The air conditioner controller is based on the following rules:

(1) Run the air conditioner when the temperature is over 78 degrees.
(2) Do not run the air conditioner when the temperature is between 78 and 81 degrees, and the AC line voltage is low (to reduce demand when the power company is heavily loaded).
(3) Never run the air conditioner if it was turned off less than four minutes ago (to protect the compressor).
(4) Do not run the air conditioner unless the other rules say to do so.[8]

These rules seem to be perfectly reasonable, but in classical logic they are potentially contradictory. For example, suppose that the temperature is 79 degrees and the line voltage is low. Then rule (1) says to turn the air conditioner on, and rule (2) says no. It is easy to see that there are other potential conflicts, too.

As already stated, contradictions in classical logic lead to trivialisation which can have serious consequences in systems control. On the other hand, defeasible logic can deal with such conflicts, as outlined in section 2. In this particular case, rules (1), (2) and (4) are defeasible, while rule (3) is strict.

Given the small size of the PIC microcontroller, it doesn't run a defeasible logic inference engine (such as d-Prolog). Instead, the system was *compiled* into a table. This was done by running d-Prolog using the above rules, together with varying sets of input values. Then the table was actually implemented on the microcontroller.

We believe that this makes this application so interesting. Defeasible logic was used *as a high level specification language* in which the properties of the air conditioner controller were specified. Of course, one could have just created the table. But that misses the point in the same way as the argument that modern programming languages are not needed, since everything can be programmed in an assembler language – even on the 0-1 level. The specification is simple, easily understandable, and moreover open to formal validation and further refinement. In fact, Covington (1997) mentions that the initial specification didn't include rule (4), and that the system gave feedback that a rule may be missing.

[8] Modelled as a rule "Don't run" which has lower priority than the rules (1) and (3).

6.3 Default reasoning in legal decision support

Legal reasoning is a domain which has attracted a lot of work seeking to apply artificial intelligence techniques (e.g. Sergot et al., 1986). This may have been caused by the superficial observation that the texts of laws can be easily represented as rules. Unfortunately, most of the early applications failed because they missed important points. For example, issues related to *open texture* (Hart, 1958), i.e., the interpretation of legal terms occurring in laws and regulations.

In the following, we will concentrate on a different point: Most early attempts tried to model laws using rules in the classical sense (that means, using material implication). But as Rissland (1988) points out, the status of legal rules "is more like that of heuristics than of theorems, in the sense that the joining of antecedents and conclusions is not ironclad." And Dewitz et al. (1994) add: "Furthermore, using material implication to represent legal rules suggests that law is immutable, that a conclusion once drawn cannot be altered. But legal rules are neither ironclad nor immutable." Here we have a clear indication that the representation of legal rules should be nonmonotonic. What is more natural than to represent legal rules using defaults? Such work has been reported in several publications (e.g., Causey, 1994; Dewitz et al., 1994).

Dewitz et al. (1994) describe a system of legal reasoning based on d-Prolog, an extension of Prolog which implements defeasible logic. The example used to test the system was the Uniform Commercial Code (UCC), Article 9. It is the main body of statutory law governing secured transactions in the United States. Secured transactions are typically sales of personal property in which the buyer uses credit. A typical rule in this domain looks as follows:

$$contract(X) \wedge part(X, Y) \wedge minor(Y) \leftarrow : enforceable(X)$$

The rule says that contracts involving students are usually not enforceable. The defeasible rules can indeed get quite complicated. Of course, in many cases there exist legal rules which substantiate contradictory conclusions. In such cases the *superiority relation* in defeasible logic proves to be very helpful. It allows one to defined preferences among rules. One simple guideline is that more specific rules override more general rules. For example, we might have the following defeasible rules:

$$contract(X) \wedge party(X, Y) \wedge minor(Y) \leftarrow : enforceable(X) \quad (1)$$

$$contract(X) \wedge party(X, Y) \wedge minor(Y) \wedge educationalLoanRepayment(X) \leftarrow enforceable(X) \quad (2)$$

Rule (1) says that contracts involving minors are usually not enforceable. Rule (2) say that contracts regulating the repayment of educational loans are enforceable. In case both rules are applicable (which would lead to contrary conclusions), rule (2) is preferred over (1) because it is more specific.

In legal reasoning there exist forms of priority information other than specificity, for example based on authority: *Lex Superior* states that a law of higher authority overrides a law of lower authority (for example, federal law overrides state law). Another legal principle is *Lex Posterior*: A more recent law overrides an older one. These and other features of legal reasoning can be naturally represented using default (defeasible) rules. Both the system of Dewitz et al. (1994) and EVID (Causey, 1994) have been successfully used.

An idea worth following in legal reasoning is the following: It would be interesting for lawyers to prepare their cases using a legal decision support system. Then they could check the facts of their cases, and design strategies that can be used in court to counter arguments of the opposite side. For example: how can I defeat this argument? Or: If the opposite side tries to attack this argument of mine, how can I counterattack?

7 Conclusion

In an invited talk at the *IJCAI'95 Workshop on Applications and Implementations of Nonmonotonic Reasoning Systems*, Hector Levesque reported about a survey among his students at the University of Toronto regarding their ranking of the difficulty of several nonmonotonic reasoning formalisms.

At first sight, it was surprising to see default logic being perceived as the second most difficult one, surpassed only by the full version of Circumscription (McCarthy, 1980), but well ahead Autoepistemic Logic (Moore, 1985). But later on he revealed that default logic had been taught along the lines of Reiter's presentation where one has to make guesses, whilst for autoepistemic logic the students had been provided with a clear operational method for determining expansions.

The basis of this tutorial paper were operational interpretations of some variants of default logic. These methods require no guessing, rather they can be applied in a straightforward way to concrete situations. It is hoped that this presentation can change the perception of default logic as a difficult, mysterious method of reasoning, and can help make the material more easily accessible to a broader audience.

Implementation aspects were outside the scope of this paper; some entry points to current work in the area include Cholewinski (1994), Cholewinski et al. (1995), Courtney et al. (1995), Linke & Schaub (1995), Niemela (1995), Risch & Schwind (1994) and Schaub (1995). The same is true for questions of computational complexity (see Gottlob, 1992 and Kautz & Selman, 1989) and applications.

References

Antoniou, G, 1997. *Nonmonotonic Reasoning* MIT Press.

Balzer, R, 1991. "Tolerating inconsistency" *Proc. 13th International Conference on Software Engineering* IEEE Press, 158–165.

Besnard, P, 1989. *An Introduction to Default Logic* Springer-Verlag.

Boehm, B and In, H, 1996. "Identifying quality requirement conflicts" *IEEE Software* March, 25–35.

Bondarenko, A, Dung, PM, Kowalski, RA and Toni, F, 1997. "An abstract, argumentation-theoretic approach to default reasoning" *Artificial Intelligence* **93**(1–2) 63–101.

Brewka, G, 1991. "Cumulative default logic: in defense of nonmonotonic inference rules" *Artificial Intelligence* **50**(2) 183–205.

Brewka, G, 1994. "Reasoning about priorities in default logic" *Proc. of the 12th National Conference on Artificial Intelligence (AAAI-94)* AAAI/MIT Press, 940–945.

Cadoli, M, Donini, FM and Schaerf, M, 1994. "Is intractability of non-monotonic reasoning a real drawback?" *Proc. 12th National Conference on Artificial Intelligence (AAAI–94)* 946–951.

Causey, RL, 1994. "EVID: A system for interactive defeasible reasoning" *Decision Support Systems* **11**.

Cholewinski, P, 1994. "Stratified default logic" *Proc. Computer Science Logic 1994, Springer LNCS 933* 456–470.

Cholewinski, P, Marek, W, Mikitiuk, A and Truszczynski, M, 1995. "Experimenting with default logic" *Proc. International Conference of Logic Programming* MIT Press.

Courtney, A, Antoniou, G and Foo, NY, 1996. "Exten: A system for computing default logic extensions" *Proc. 4th Pacific Rim International Conference on Artificial Intelligence (PRICAI-96): Springer LNAI 1114* 471–482.

Covington, MA, 1997. "Defeasible logic on an embedded microcontroller" *Proc. 10th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* June.

Delgrande, JP, Schaub, T and Jackson, WK, 1994. "Alternative approaches to default logic" *Artificial Intelligence* **70** 167–237.

Dewitz, SK, Ryu, Y and Lee, RM, 1994. "Defeasible reasoning in law" *Decision Support Systems* **11** 133–155.

Etherington, D, 1987a. "Formalizing nonmonotonic reasoning systems" *Artificial Intelligence* **31** 41–85.

Etherington, D, 1987b. *Reasoning with Incomplete Information* Pitman.

Gardenfors, P, 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States* MIT Press.

Gardenfors, P and Makinson, D, 1994. "Nonmonotonic inference based on expectations" *Artificial Intelligence* **65** 197–245.

Gelfond, M, Lifschitz, V, Przymusinska, H and Truszczynski, M, 1991. "Disjunctive Defaults" *Proc. 2nd International Conference on Principles of Knowledge Representation and Reasoning* Morgan Kaufmann, 230–237.

Gottlob, G, 1992. "Complexity results for nonmonotonic logics" *Journal of Logic and Computation* **2** 397–425.

Grosof, B, 1991. "Generalizing prioritization" *Proc. 2nd International Conference on Principles of Knowledge Representation and Reasoning* Morgan Kaufmann, 289–300.

Grosof, B, 1996. *"Practical Prioritized Defaults via Logic Programs"* IBM Research Report RC 20464, T.J. Watson Research Center.

Hart, HLA, 1958. "Positivism and the separation of law and morals" *Harvard Law Review* **71** 593–629.

Hunter, A and Nuseibeh, B, 1997. "Analysing inconsistent specifications" *Proc. Third International Symposium on Requirements Engineering* IEEE Press.

Kautz, HA and Selman, B, 1989. "Hard problems for simple default logics" *Proc. 1st International Conference on Principles of Knowledge Representation and Reasoning* Morgan Kaufmann, 189–197.

Linke, T and Schaub, T, 1995. "Lemma handling in default logic theorem proving" *Proc. Symbolic and Quantitative Approaches to Reasoning and Uncertainty: Springer, LNAI 946* 285–292.

Lukaszewicz, W, 1988. "Considerations on default logic" *Computational Intelligence* **4** 1–16.

Lukaszewicz, W, 1990. *Non-Monotonic Reasoning – Formalization of commonsense reasoning* Ellis Horwood.

Makinson, D, 1994. "General patterns in nonmonotonic reasoning" In: *Handbook of Logic in Artificial Intelligence and Logic Programming Vol. 3* Oxford University Press, 35–110.

Marek, W and Truszczynski, M, 1993. *Nonmonotonic Logic*. Springer-Verlag.

McCarthy, J, 1980. Circumscription – A form of non–monotonic reasoning" *Artificial Intelligence* **13** 27–39.

Mikitiuk, A and Truszczynski, M, 1995. "Constrained and rational default logics" *Proc. 14th International Joint Conference on Artificial Intelligence* Morgan Kaufmann, 1509–1515.

Moore, RC, 1985. "Semantical considerations on non-monotonic logic" *Artificial Intelligence* **25** 75–94.

Niemela, I, 1995. "Towards efficient default reasoning" *Proc. 14th International Joint Conference on Artificial Intelligence* Morgan Kaufmann, 312–318.

Nuseibeh, B, 1996. "To Be And Not To Be: On managing inconsistency in software development" *Proc. 8th International Workshop on Software Specification and Design* IEEE Press, 164–169.

Nute, D, 1987. "Defeasible reasoning" *Proc. 20th Hawaii International Conference on Systems Science* IEEE Press, 470–477.

Nute, D, 1994. "Defeasible logic" DM Gabbay, CJ Hogger and JA Robinson eds, *Handbook of Logic in Artificial Intelligence and Logic Programming Vol 3* Oxford University Press, 353–395.

Poole, D, 1994. "Default Logic" In: *Handbook of Logic in Artificial Intelligence and Logic Programming* Oxford University Press.

Reiter, R, 1978. "On closed-world data bases" In H Gallaire and J Minker eds, *Logic and Data Bases* Plenum Press, 55–76.

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

Risch, V and Schwind C, 1994. "Tableau-based characterization and theorem proving for default logic" *Journal of Automated Reasoning* **13** 223–242.

Rissland, EL, 1988. "Artificial intelligence and legal reasoning: A discussion of the field and Gardner's book" *AI Magazine* Fall, 45–55.

Schaub, T, 1992. "On constrained default theories" *Proc. 10th European Conference on Artificial Intelligence* Wiley, 304–308.

Schaub, T, 1995. "A new methodology for query-answering in default logics via structure-oriented theorem proving" *Journal of Automated Reasoning* **15**(1) 95–165.

Sergot, MJ, Sadri, R, Kowalski, RA, Kriwaczek, F, Hammond, P and Cory, HT, 1986. "The British Nationality Act as a Logic Program"*Communications of the ACM* **29** 370–386.

Teng, CM, 1996. "Possible world partition sequences: A unifying framework for uncertain reasoning" *Proc. 12th Conference on Uncertainty in Artificial Intelligence* 517–524.

Touretzky, DS, Horty, JF and Thomason, RH, 1987. "A clash of intuitions: The current state of nonmonotonic multiple inheritance systems" *Proc. 10th International Joint Conference on Artificial Intelligence* MIT Press.

Zowghi, D, Ghose, AK and Peppas, P, 1996. "A framework for reasoning about requirements evolution" *Proc. Fourth Pacific Rim International Conference on Artificial Intelligence: Springer LNAI 1114* 157–168.

Zowghi, D and Offen, R, 1997. "A logical framework for modeling and reasoning about the evolution of requirements" *Proc. Third International Symposium on Requirements Engineering* IEEE Press.