Nichtmonotones Schließen

Gerhard Brewka
Universität Leipzig, Institut für Informatik
Augustusplatz 10-11, D-04109 Leipzig

Zusammenfassung

In diesem Abschnitt wird zunächst die Rolle des nichtmonotonen Schließens für die Künstliche Intelligenz erörtert. Nach der Darstellung einiger einfacher Formen der Nichtmonotonie, wie sie in verschiedenen weit verbreiteten Systemen vorkommt, stellt Abschnitt 2 einige der wichtigsten existierenden nichtmonotonen Logiken dar: Reiters Default Logik, Moores Autoepistemische Logik und McCarthys Circumscription. Abschnitt 3 beschreibt ausführlich einen Ansatz, der Default-Schließen auf inkonsistenztolerantes Schließen zurückführt. Dabei werden bevorzugte maximal konsistente Teilmengen der Prämissen berücksichtigt. Es wird gezeigt, wie durch eine geeignete Definition der bevorzugten Teilmengen Prioritäten zwischen Default-Regeln auf einfache Weise ausgedrückt werden können. Abschnitt 4 geht auf den Zusammenhang zwischen nichtmonotonen Logiken und Logikprogrammierung ein und diskutiert nichtmonotone Inferenzsysteme. Abschnitt 5 beschreibt verschiedene Anwendungen des nichtmonotonen Schließens.

1 Einführung

Unser empirisches Wissen über die Welt ist stets unvollständig, und sehr häufig lassen sich aus ihm nicht alle Fakten ableiten, die wir benötigen, um darauf Entscheidungen, Planungen und Aktionen zu gründen. Trotz solcher Lücken in unserem Wissen sind wir natürlich gezwungen zu handeln. Vielfach verwenden wir deshalb Regeln mit Ausnahmen, um das fehlende Wissen zu ergänzen. Solche Regeln drücken aus, was typischerweise der Fall ist, und können dazu benutzt werden, plausible Konklusionen abzuleiten, sofern keine der Konklusion widersprechende Information vorliegt. Jeder intelligente Agent muß in der Lage sein, solche Regeln – im KI-Deutsch werden sie auch Default-Regeln genannt – sinnvoll zu handhaben.

Eine zweite für uns wichtige Fähigkeit ist der Umgang mit inkonsistenter Information. Es kommt ständig vor, daß sich die uns vorliegende Information widerspricht, z.B. weil sie aus unterschiedlichen Quellen stammt. Natürlich dürfen wir uns dadurch nicht paralysieren lassen. Wir müssen auf sinnvolle Weise mit der inkonsistenten Information umgehen.

Für beide Probleme bietet die klassische Prädikatenlogik für sich genommen keine Lösung: im Falle inkonsistenter Prämissen liefert die Prädikatenlogik die Menge aller Formeln als Theoreme. Das ist sicher nicht das, was wir wollen, denn wer auf widersprüchliche Information stößt, wird sicher nicht anfangen, auf einmal alle beliebigen Sätze für wahr zu halten. Auch die Behandlung von Regeln mit Ausnahmen läßt sich in der klassischen Logik nicht adäquat modellieren. Natürlich können wir eine Regel wie *Vögel fliegen typischerweise* folgendermaßen darstellen:

$$\forall x.Vogel(x) \land \neg Ausnahme(x) \Rightarrow Fliegt(x)$$

$$\forall x.Ausnahme(x) \Leftrightarrow Pinguin(x) \lor Strauß(x) \lor \neg HatFlügel(x) \ldots$$

Diese Darstellung erfordert aber eine vollständige Auflistung aller möglichen Ausnahmen, eine für sich genommen unmögliche Aufgabe, denn auch das Wissen um Ausnahmen ist natürlich so gut wie immer unvollständig. Aber selbst wenn eine solche vollständige Liste verfügbar wäre, wäre die Darstellung immer noch inadäquat: um für einen bestimmten Vogel, sagen wir Tweety, abzuleiten, daß er fliegt, ist es notwendig zu beweisen, daß kein Ausnahmefall vorliegt, d.h. daß Tweety kein Pinguin ist, kein Strauß, daß er Flügel hat, usw. Das aber wollen wir gerade nicht: wir wollen *Tweety fliegt* ableiten können, wenn nicht gezeigt werden kann, daß Tweety eine Ausnahme ist, nicht nur dann, wenn gezeigt werden kann, daß er keine Ausnahme ist.

Worin ist diese Schwierigkeit begründet? Die klassische Logik besitzt folgende Monotonieeigenschaft: Für alle Mengen von Prämissen A und Formeln p, q gilt:

$$A \vdash q => A \cup \{p\} \vdash q$$

d.h. zusätzliche Information kann nie alte Konklusionen ungültig machen ($\vdash$ bezeichnet die klassische Ableitbarkeitsrelation). Wie wir im obigen Beispiel gesehen haben, wollen wir eine Default-Regel dazu benutzen, um eine plausible Konklusion abzuleiten, sofern nichts auf einen Ausnahmefall hindeutet. Wenn wir später zusätzliche Information erhalten, die besagt, daß entgegen unseren Erwartungen doch eine Ausnahme vorliegt, dann muß diese Konklusion zurückgenommen werden. Jedes Schließen, das auf Regeln mit Ausnahmen basiert, muß deshalb in diesem Sinne nichtmonoton sein. Dasselbe gilt konsequenterweise auch für eine Logik, die solches Schließen formalisiert.

Uns geht es hier vor allem um Default-Schließen. Es gibt aber andere Formen nichtmonotonen Schließens, die wenigstens erwähnt werden sollen. Wie gesagt erlauben Default-Regeln die Ableitung plausibler Konklusionen aus unvollständigem Wissen, sofern keine widersprüchliche Information vorliegt. Eine ganz andere Form nichtmonotonen Schließens ist das autoepistemische Schließen. Dabei wird Wissen über das eigene Wissen (autoepistemisches Wissen) zur Ableitung korrekter Schlüsse verwendet. Das Standardbeispiel lautet:

> *Ich kenne alle meine Brüder.*
> *(= Wenn x mein Bruder ist, so weiß ich das.)*
> *Ich habe keine Information, daß Peter mein Bruder ist.*
> *Also ist Peter nicht mein Bruder.*

Die Nichtmonotonie entsteht in diesem Fall deshalb, weil die Bedeutung von Aussagen über das eigene Wissen kontextabhängig ist: wenn wir in unserem Beispiel zusätzliche Information erhalten, die besagt, daß Peter doch mein Bruder ist, so wissen wir, daß die erste Prämisse falsch war, als sie zur Ableitung von *Peter ist nicht mein Bruder* verwendet wurde. Aber das heißt nicht, daß diese Prämisse nun aufgegeben werden muß. Die Prämisse bezieht sich auf das jeweils aktuelle Wissen, und es ist durchaus sinnvoll anzunehmen, daß sie bezogen auf das jetzt erweiterte Wissen zutrifft. Eine genauere Analyse des autoepistemischen Schließens findet sich in [Moore, 85].

Auch das Schließen auf der Basis inkonsistenter Information (inkonsistenztolerantes Schließen) ist nichtmonoton. Betrachten wir die inkonsistente Prämissenmenge

$$\{p, \neg p, q, r, s\}$$

Wie würde ein intelligenter Agent mit solcher Information umgehen? Er würde sicherlich nicht alle verfügbare Information als gleichermaßen wertlos ansehen. Eine Möglichkeit wäre es, die maximalkonsistenten Teilmengen der Prämissen zu betrachten und als ableitbar anzusehen, was aus all diesen Teilmengen abgeleitet werden kann. In unserem Beispiel hätte das den Effekt, daß p und $\neg p$ 'eliminiert' würden und nur die anderen Prämissen für Ableitungen verwendet werden könnten. Natürlich kann dabei das Zufügen von neuer Information, etwa $\neg q$, weitere Prämissen unwirksam machen. Damit ist auch diese Form von Schließen nichtmonoton. Wir werden in Abschnitt 3 noch einmal ausführlicher auf inkonsistenztolerantes Schließen zurückkommen.

Bevor wir uns einigen wichtigen Formalisierungen zuwenden, wollen wir noch einige Beispiele diskutieren, die die Relevanz nichtmonotonen Schließens für die Künstliche Intelligenz zeigen. Historisch gesehen wurden die Arbeiten auf diesem Gebiet vor allem motiviert durch das berühmte Frame-Problem: Wie läßt sich adäquat repräsentieren, daß die meisten Objekte sich nicht ändern, wenn ein Ereignis eintritt? Das Problem trat vor allem im Zusammenhang mit dem Situationskalkül deutlich zutage. Der Situationskalkül wurde von McCarthy und Hayes entwickelt, um in einem logischen Rahmen das Schließen über Aktionen und Ereignisse in der Zeit zu modellieren. Damit die Effekte von Ereignissen dargestellt werden können, werden Fakten mit den Situationen indiziert, in denen sie gelten, etwa:

$$Holds(In(Fred, Kitchen), Sit105)$$

$$Holds(Color(Kitchen, Red), Sit105)$$

Ereignisse produzieren neue Situationen, etwa:

$$Sit106 = Result(Go(Fred, Bathroom), Sit105)$$

Weitere Axiome beschreiben, wie Ereignisse die Welt verändern:

$$\forall x, y, s. Holds(In(x, y), Result(Go(x, y), s)$$

Das Problem ist nun, z.B. zu zeigen, welche Farbe die Küche in $Sit106$ hat. Um ableiten zu können, daß sie immer noch rot ist, brauchen wir folgendes Axiom:

$$\forall x, y, v, w, s. Holds(Color(x, y), s) \Rightarrow Holds(Color(x, y), Result(Go(x, y), s))$$

Solche *Frame-Axiome* sind für jedes Paar bestehend aus einem Ereignis und einem durch dieses Ereignis nicht veränderten Faktum nötig. Das macht den Ansatz für die Praxis völlig unbrauchbar. Eine mögliche Lösung dieses Problems wäre die Einführung eines *Persistenz-Defaults*, das informell etwa so lauten könnte:

Ereignisse verändern Eigenschaften normalerweise nicht.

Mit einem solchen Default, so die Grundidee, ist nur noch die Beschreibung der tatsächlichen Effekte von Ereignissen erforderlich. Explizite Frame-Axiome erübrigen sich. Ihre Konklusionen liefert das Persistenz-Default.

Mittlerweile hat sich allerdings herausgestellt, daß eine einfache Repräsentation dieses Defaults in einer der inzwischen entwickelten nichtmonotonen Logiken oft nicht zu den erwarteten Resultaten führt. Eine Analyse der dabei auftretenden Probleme und eine Darstellung einiger Lösungsversuche finden sich in [Hanks und McDermott, 87].

Ein verwandtes Problem ist das Qualifikationsproblem: wie läßt sich adäquat repräsentieren, daß Aktionen fehlschlagen können? Wenn man z.B. den Autoschlüssel im Zündschloß dreht, so ist der erwartete Effekt, daß der Motor anspringt. Aber, jeder kennt das, es gibt Ausnahmesituationen, in denen dieser Effekt nicht eintritt: der Tank kann ausgelaufen sein, die Batterie leer, die Zündkerzen defekt, der Motor ausgebaut, usw. Wieder ist die Liste der möglichen Ausnahmen beliebig verlängerbar. Keiner von uns wird wohl jedesmal nachsehen, ob der Motor noch da ist, bevor er zu starten versucht. Unsere Handlungen basieren auf der Erwartung, daß sich die Dinge wie üblich verhalten. Wieder ist es eine natürliche Idee, Defaults zu verwenden, um die erwarteten Effekte von Ereignissen formal zu beschreiben.

Es gibt zahlreiche weitere Gebiete, in denen Nichtmonotonie eine wichtige Rolle spielt. Bei der Diagnose etwa können Defaults verwendet werden, um das normale Verhalten von Teilen eines Gerätes zu beschreiben. Fügt man zu dieser Beschreibung des normalen Verhaltens die Beschreibung des tatsächlich beobachteten Verhaltens hinzu, so geben die nicht angewendeten Defaults Auskunft darüber, welche Bauteile defekt sein können. Beim Sprachverstehen verwenden wir ständig Defaults, etwa um Mehrdeutigkeiten zu beheben. Beim Bilderkennen werden Defaults benutzt, um aus Teilszenen vollständige Bilder zu erzeugen. Selbst das Gesetz ist nichtmonoton, wie folgende Regeln zeigen, die in dieser oder ähnlicher Form tatsächlich in Gesetzestexten zu finden sind [Gordon, 87]:

Verträge sind gültig.
Verträge mit Minderjährigen sind ungültig.
Verträge mit Minderjährigen, die im Beisein eines Vormundes geschlossen werden, sind gültig.

Es ist leicht zu sehen, daß in all diesen Fällen zusätzliche Information zur Rücknahme von Schlüssen führen kann.

Nichtmonotone Systeme haben eine recht lange Tradition in der Künstlichen Intelligenz. Schon sehr lange werden etwa Frame-Systeme verwendet, wie sie derzeit in fast allen kommerziellen Expertensystemwerkzeugen zu finden sind. Diese Systeme ermöglichen die Beschreibung von Klassenhierarchien und typischen Eigenschaften von Instanzen dieser Klassen. Frames beschreiben Klassen, Slots repräsentieren Attribute der Instanzen dieser Klassen und ihre typischen Werte. Die grundlegende Idee ist die, daß im Falle von Widersprüchen die spezifischste Information vorgezogen wird. Betrachten wir folgendes Beispiel (die hier für Frame- und Instanzdefinitionen verwendete Sprache stammt aus dem in der GMD entwickelten BABYLON-System und dürfte selbsterklärend sein):

```
(defframe auto
     (slots  (sitze 5)
             (zylinder 4)
             (raeder 4)))

(defframe sportwagen
     (supers auto)
     (slots  (sitze 2)
             (preis hoch)))

(definstance speedy of sportwagen)
```

Aus diesen Definitionen läßt sich ableiten, daß `speedy` 2 Sitze und 4 Zylinder hat. Der aus der Definition von `auto` stammende Wert 5 für `sitze` wird nicht berücksichtigt, da `auto` eine Oberklasse von `sportwagen` ist, wie es durch die entsprechende supers-Spezifikation festgelegt wurde. Wir erweitern nun die zweite Definition folgendermaßen:

```
(defframe sportwagen
     (slots   (sitze 2)
              (preis hoch)
              (zylinder 6)))
```

Jetzt wird für `zylinder` der Wert 6 abgeleitet, d.h. zusätzliche Information hat zur Revision einer früheren Ableitung geführt.

Ein anderes bekanntes nichtmonotones System ist Prolog mit seiner Behandlung der Negation (negation as failure). Dabei wird *not* A als gültig betrachtet, wenn A nicht abgeleitet werden kann. Betrachten wir wieder unser altes Tweety-Beispiel:

```
Fliegt(_x) :- Vogel(_x), not Abnormal(_x).
Abnormal(_x) :- Pinguin(_x).
Vogel(Tweety).
```

Das Prolog-Ziel `Fliegt(Tweety)`? liefert SUCCESS, da `not Abnormal(Tweety)` abgeleitet werden kann. Fügen wir jedoch die Prämisse

```
Pinguin(Tweety).
```

hinzu, so kann `Fliegt(Tweety)` nicht mehr abgeleitet werden. Auch hier also ein Fall von Nichtmonotonie.

Die vor allem aus dem Bereich der deduktiven Datenbanken bekannte Closed World Assumption (CWA) ist in engem Zusammenhang hierzu zu sehen. Ähnlich wie in Prolog geht man davon aus, daß nur positive Information explizit gespeichert werden muß. Fehlen einer Information wird als Falschheit dieser Information gedeutet. Wenn etwa in einer Datenbank über Flugverbindungen keine Direktverbindung zwischen Bonn und London enthalten ist, so wird das so interpretiert, daß es eben keine solche Verbindung gibt. Formal läßt sich die CWA folgendermaßen präzisieren:

Definition 1 *Sei T eine Menge von Formeln. p ist ableitbar aus T unter der CWA genau dann wenn*

$$T \cup ASS(T) \vdash p$$

wobei $ASS(T) := \{\neg q \mid q \text{ ist atomar und es gilt } T \not\vdash q\}$.

Wie diese drei Beispiele zeigen, spielen nichtmonotone Systeme schon lange eine wichtige Rolle in der Künstlichen Intelligenz und, allgemeiner, in der Informatik. Allerdings sind die Formen der Nichtmonotonie, die wir hier vorgestellt haben, äußerst beschränkt und lassen sich nicht einfach verallgemeinern. So führt etwa die CWA zu Inkonsistenz, wenn Disjunktionen in den Prämissen zugelassen werden. Seit Ende der achtziger Jahre hat sich das nichtmonotone Schließen als eigenständiges Forschungsgebiet der KI etabliert. Vorrangiges Ziel war zunächst, geeignete Formalisierungen zu finden, die mindestens so ausdrucksstark sind wie die die klassische Logik. Einige wichtige Ergebnisse dieser Bemühungen werden im folgenden Abschnitt dargestellt.

2 Formalisierungen nichtmonotonen Schließens

Bevor wir in diesem Abschnitt einige der wichtigsten nichtmonotonen Logiken darstellen, wollen wir kurz ein grundlegendes Problem diskutieren, das jede Formalisierung technisch schwierig macht: das Problem sich widersprechender Defaults. Hier das Standardbeispiel:

Quäker sind Pazifisten.
Republikaner sind keine Pazifisten.
Nixon ist Quäker und Republikaner.

Ist Nixon Pazifist oder nicht? Es gibt zwei Defaults, die jeweils benutzt werden können, um sich widersprechende Konklusionen abzuleiten. Die Defaults sind für sich genommen völlig korrekt, sie können aber natürlich nicht beide gleichzeitig verwendet werden. Es scheint in diesem Fall unterschiedliche Mengen von Überzeugungen zu geben, die durch die Defaults gleichermaßen gestützt werden. Wie wir sehen werden, tragen einige Ansätze dem dadurch Rechnung, daß sie verschiedene Formelmengen, sogenannte Extensionen, generieren, in denen jeweils eine maximale Menge von Defaults angewendet wurde. Damit stellt sich natürlich die Frage, was eigentlich die ableitbaren Formeln sind. Es gibt zwei Möglichkeiten: einer skeptischen Sichtweise entspricht es, nur das zu glauben, was in allen Extensionen gilt. Diese Sichtweise ist, wie wir sehen werden, implizit in McCarthys Circumscription. Reiter dagegen vertritt die Sichtweise, daß jede der erzeugten Extensionen für sich genommen als akzeptable Menge von Überzeugungen eines Agenten aufgefaßt werden kann.

Zu den wichtigsten Ansätzen der Formalisierung nichtmonotonen Schließens gehören:

- Default-Logik (Reiter). Nichtklassische Inferenzregeln werden verwendet, um Defaults darzustellen. Die Extensionen werden als Fixpunkte eines Operators definiert. Der Operator garantiert, daß in einer Extension möglichst viele Defaults angewendet wurden.

- Der modale Ansatz (McDermott und Doyle, Moore). Hier drückt ein Modaloperator explizit aus, ob etwas geglaubt wird oder konsistent ist. Wiederum wird ein Fixpunktoperator verwendet, um die Extensionen zu definieren.

- Circumscription (McCarthy, Lifschitz). Hier wird Folgerbarkeit nicht, wie üblich, als Gültigkeit in allen, sondern als Gültigkeit in bestimmten, bevorzugten Modellen der Prämissen definiert. Syntaktisch werden die 'uninteressanten' Modelle durch Hinzufügen eines Formelschemas bzw. einer Formel 2. Stufe eliminiert.

- Konditionale Ansätze (Delgrande, Kraus/Lehmann/Magidor, Boutilier). Sie verwenden ein Konditional $\rightarrow$, also einen nicht-wahrheitsfunktionalen Junktor, um Defaults in der logischen Sprache zu repräsentieren. Eine Formel q folgt nichtmonoton aus einer Formel p, wenn das Konditional $p \rightarrow q$ in dem entsprechenden Ansatz ableitbar ist.

Wir werden einige dieser Ansätze im folgenden näher erläutern. Dabei werden wir die Default Logik ausführlich beschreiben. Die Darstellung der autoepistemischen Logik sowie der Circumscription beschränkt sich auf die wichtigsten Grundideen. Eine Darstellung der konditionalen Ansätze würde den Rahmen dieses Kapitels sprengen. Als Einstieg hierzu empfehlen wir dem Leser [Delgrande, 88, Kraus et al., 90, Boutilier, 94].

2.1 Default Logik

In der Default Logik (DL) [Reiter, 80] werden Default-Regeln als eine Art von nichtklassischen Inferenzregeln repräsentiert. Eine Default-Theorie ist ein Paar (D, W), wobei W, eine Menge von Formeln 1. Ordnung, das sichere Wissen repräsentiert. D ist eine Menge von Defaults der Form

$$\frac{A : B_1, \ldots, B_n}{C}$$

wobei A, B_i und C klassische Formeln sind (als alternative Notation wird häufig auch $A{:}B_1, \ldots, B_n/C$ verwendet). Diese Default-Regel ist intuitiv in folgendem Sinne zu lesen: wenn A ableitbar ist und, für alle i $(1 \leq i \leq n)$, $\neg B_i$ nicht abgeleitet werden kann, dann leite C ab. A heißt Vorbedingung, B_i Konsistenzannahme und C Konsequenz des Defaults. Wir werden im folgenden offene Defaults, d.h. Defaults mit freien Variablen, als Schemata interpretieren, die alle ihre Grundinstanzen repräsentieren.

Die Frage ist nun: gegeben eine Default-Theorie (D, W), was sind die durch sie induzierten akzeptablen Überzeugungsmengen, die Extensionen genannt werden? Es gibt einige Eigenschaften, die eine solche Menge S erfüllen sollte:

1. sie sollte das sichere Wissen W enthalten,

2. sie sollte im Sinne der klassischen Logik abgeschlossen sein,

3. alle 'anwendbaren' Defaults sollten angewendet worden sein, wobei Anwendbarkeit in Bezug auf S selbst definiert werden muß,

4. sie sollte keine Formel enthalten, die sich nicht aus W zusammen mit den Konsequenzen 'anwendbarer' Defaults in D herleiten läßt.

Die ersten drei Eigenschaften lassen sich direkt formal aufschreiben, etwa in Form der folgenden Definition:

Definition 2 *Sei (D, W) eine Default-Theorie. S ist bezüglich (D, W) abgeschlossen gdw.*

1. $W \subseteq S$,

2. $Th(S) = S$

3. falls $A{:}B_1, \ldots, B_n/C \in D$, $A \in S$, $\neg B_i \notin S$ $(1 \leq i \leq n)$, *dann* $C \in S$.

Allerdings trägt die Abgeschlossenheit einer Menge von Formeln der vierten gewünschten Eigenschaft noch nicht Rechnung. Eine in der Mathematik und Informatik übliche Art, unerwünschte Elemente aus Mengen auszuschließen, ist die Forderung nach Minimalität von Mengen. Es liegt also nahe, Extensionen als kleinste Mengen zu definieren, die die obige Definition erfüllen.

Leider hilft das in unserem Fall nicht weiter. Betrachten wir die simple Default-Theorie $(\{true{:}b/a\}, \emptyset)$. Zwar erfüllt, wie gewünscht, die Menge $S_1 = Th(\{a\})$ unsere obige Definition, aber leider auch die Menge $S_2 = Th(\{\neg b\})$. Es ist leicht zu sehen, daß die drei Eigenschaften von Definition 2 gelten, außerdem gibt es keine echte Teilmenge von S_2, die diese Bedingungen erfüllt, d.h. S_2 ist auch minimal. Das zeigt, daß wir unsere vierte gewünschte Bedingung auf diesem Weg nicht erreichen können.

Reiter verwendet deshalb einen Trick. Er testet einen Kandidaten, also eine Menge von Formeln S, auf folgende Weise: S wird überführt in eine Menge $\Gamma(S)$. $\Gamma(S)$ ist die kleinste Menge, die die drei ersten Eigenschaften aus unserer obigen Aufzählung erfüllt, wobei jedoch die Anwendbarkeit einer Default-Regel bezüglich S (und nicht $\Gamma(S)$) festgestellt wird. S hat den Test bestanden, ist also Extension, wenn es sich bei diesem Test reproduziert, wenn also $\Gamma(S) = S$. Es ist sofort zu sehen, daß nur solche Mengen diesen Test bestehen, die bezüglich (D, W) abgeschlossen sind. Aber auch Eigenschaft 4 muß erfüllt sein: wenn ein Kandidat Formeln enthält, die nicht aus W oder durch Anwendung von Defaults hergeleitet werden können, so werden diese bei dem Test nicht reproduziert, und der falsche Kandidat scheidet aus. In unserem Beispiel etwa ergibt $\Gamma(S_2)$ die Menge aller Tautologien. Damit ist S_2 keine Extension.

Hier nun die endgültige Fassung von Reiters Definition der Extensionen:

Definition 3 *Sei* (D, W) *eine Default-Theorie,* S *eine Menge von Formeln. Wir definieren einen Operator* Γ, *so daß* $\Gamma(S)$ *die kleinste Menge ist, für die gilt:*

1. $W \subseteq \Gamma(S)$,

2. $Th(\Gamma(S)) = \Gamma(S)$

3. falls $A{:}B_1, \ldots, B_n/C \in D$, $A \in \Gamma(S)$, $\neg B_i \notin S$ $(1 \leq i \leq n)$, *dann* $C \in S$.

E *ist eine Extension von* (D, W) *genau dann wenn* E *Fixpunkt von* Γ *ist, d.h. wenn gilt* $\Gamma(E) = E$.

Reiter hat eine äquivalente, quasi-induktive Charakterisierung der Extensionen gegeben. Diese Version wird häufig in Beweisen benutzt und macht sehr deutlich, in welchem Sinne Formeln in den Prämissen gegründet sein müssen. Sei E eine Menge von Formeln. Wir definieren für eine gegebene Default-Theorie (D, W) eine Folge von Formelmengen wie folgt:

$$E_0 = W, \text{ und für } i \geq 0$$
$$E_{i+1} = Th(E_i) \cup \{C \mid A{:}B_1, \ldots, B_n/C \in D, A \in E_i, \neg B_i \notin E\}$$

Reiter hat gezeigt, daß E eine Extension von (D, W) ist, gdw. $E = \bigcup_{i=0}^{\infty} E_i$. Das Vorkommen von E in der Definition von E_{i+1} macht diese Definition nicht-konstruktiv.

Die Tabelle 1 stellt einige einfache Beispiele dar. Das letzte Beispiel in der Tabelle zeigt, wie der Konsistenztest in einem Default verwendet werden kann, um Prioritäten zwischen Defaults auszudrücken.

Reiters Default-Logik ist heute sicher eine der prominentesten nichtmonotonen Logiken. Dafür gibt es zwei Gründe. Zum einen ist – trotz der etwas trickreichen technischen Definition der Extensionen – die der Default-Logik zugrundeliegende Idee einfach und intuitiv: verwende Inferenzregeln mit einem zusätzlichen Konsistenztest für die Darstellung von Defaults. Zweitens

D	W	Fixpunkte(e)
$\frac{Bird(x):Flies(x)}{Flies(x)}$	$Bird(Tw)$	$Th(W \cup \{Flies(Tw)\})$
$\frac{Bird(x):Flies(x)}{Flies(x)}$	$Bird(Tw)$ $Peng(Tw)$ $\forall x.Peng(x) \Rightarrow \neg Flies(x)$	$Th(W)$
$\frac{Bird(x):Flies(x)}{Flies(x)}$ $\frac{Peng(x):\neg Flies(x)}{\neg Flies(x)}$	$Bird(Tw)$ $Peng(Tw)$	$Th(W \cup \{Flies(Tw)\})$ $Th(W \cup \{\neg Flies(Tw)\})$
$\frac{Bird(x):Flies(x) \wedge \neg Peng(x)}{Flies(x)}$ $\frac{Peng(x):\neg Flies(x)}{\neg Flies(x)}$	$Bird(Tw)$ $Peng(Tw)$	$Th(W \cup \{\neg Flies(Tw)\})$

Tabelle 1: Einige einfache Beispiele für Defaults

hat sich herausgestellt, daß die Default-Logik in bestimmter Hinsicht expressiver ist als manch konkurrierender Ansatz, etwa Circumscription. Diese zusätzliche Ausdrucksmächtigkeit ist nötig, um z.B. die Semantik für logisches Programmieren erfassen zu können.

Durch die Verwendung von Inferenzregeln lassen sich vor allem Probleme mit der Kontraposition von Defaults vermeiden. In der klassischen Logik ist eine Implikation $A \Rightarrow B$ äquivalent zu ihrer Kontraposition $\neg B \Rightarrow \neg A$. Im Zusammenhang mit Defaults ist Kontraposition manchmal unerwünscht. Z.B. ist es sicher richtig, daß Informatiker normalerweise wenig über Nichtmonotonie wissen. Daraus folgt aber nicht, daß jemand, der viel über Nichtmonotonie weiß, normalerweise kein Informatiker ist. Durch die Verwendung von Inferenzregeln zur Darstellung von Defaults lassen sich solche unerwünschte Effekte vermeiden.

Allerdings hat die Verwendung von Inferenzregeln im Sinne von Reiter auch ihre Nachteile. Zunächst kann es vorkommen, daß es gar keine Extension gibt. Man betrachte etwa die Theorie

$$(\{: \neg A/A\}, \{\}).$$

Keine Menge von Formeln, die A nicht enthält, ist eine Extension, da die Default-Regel nicht angewendet wurde. Aber auch keine Menge S, die A enthält, kann eine Extension sein: wenn S die Formel A enthält, dann wird dadurch bei der Konstruktion von $\Gamma(S)$ die Default-Regel unanwendbar. Damit kann die Formel A nicht mehr in $\Gamma(S)$ enthalten sein und S ist kein Fixpunkt.

Es gibt auch Situationen, in denen intuitiv erwartete Resultate nicht erzielt werden, wie folgendes Beispiel zeigt (da Defaults und Fakten syntaktisch unterschieden werden können, lassen wir in diesem und den folgenden Beispielen D und W implizit):

1) $Italiener{:}Trinkt\_Wein/Trinkt\_Wein$
2) $Franzose{:}Trinkt\_Wein/Trinkt\_Wein$
3) $Italiener \vee Franzose$

Man würde erwarten, aus dieser Default-Theorie $Trinkt\_Wein$ ableiten zu können, denn, unabhängig davon, ob die betreffende Person Italiener oder Franzose ist, eine der Default-Regeln sollte anwendbar sein. In der Default-Logik jedoch kann ein Default nur dann angewendet werden, wenn seine Vorbedingung bereits abgeleitet wurde. Default-Logik ist deshalb nicht in der Lage, Fallunterscheidungen adäquat zu behandeln.

Angeregt durch Arbeiten von Gabbay [Gabbay, 85], Makinson [Makinson, 89] und anderen [Kraus et al., 90] ist es in jüngerer Zeit üblich geworden, dieses und ähnliche Probleme in Form von metatheoretischen Eigenschaften nichtmonotoner Inferenzrelationen zu formulieren. Die hier relevante Eigenschaft wird üblicherweise OR genannt. $\mid\!\sim$ bezeichne eine beliebige (nichtmonotone)

Inferenzrelation, d.h. eine Relation mit einer Menge von Formeln auf der linken und einer Formel auf der rechten Seite. OR läßt sich folgendermaßen formulieren:

$$OR: \text{Wenn } X \cup \{y\} \mid\!\sim a \text{ und } X \cup \{z\} \mid\!\sim a \text{ dann } X \cup \{y \vee z\} \mid\!\sim a$$

Es stellt sich hier natürlich die Frage, was die der Default-Logik entsprechende Inferenzrelation ist. Um solch eine Relation geeignet zu definieren, wird üblicherweise die Menge von Defaults D festgehalten. Mit anderen Worten, jede Menge D erzeugt ihre eigene Inferenzrelation $\vdash_D$, die auf folgende Weise definiert werden kann:

Definition 4 *Sei D eine Menge von Reiterschen Defaults, W eine Menge von Formeln erster Ordnung, p eine Formel. Wir definieren $W \vdash_D p$ gdw. p in allen Extensionen von (D, W) enthalten ist.*

Unser Weinbeispiel zeigt, daß $\vdash_D$ die Eigenschaft OR verletzt.

Es ist zu bemerken, daß auf Fallunterscheidung basierende Schlüsse möglich sind, wenn wir unsere Defaults in folgender Form repräsentieren:

$$true{:}Italiener \Rightarrow Trinkt\_Wein/Italiener \Rightarrow Trinkt\_Wein$$

Dadurch entstehen aber wiederum Probleme mit der Kontraposition, denn nun kann $\neg Italiener$ von $\neg Trinkt\_Wein$ abgeleitet werden. Eine bessere Darstellung ist deshalb:

$$true{:}Trinkt\_Wein/Italiener \Rightarrow Trinkt\_Wein$$

Jetzt können wir die Default-Regel nicht mehr benutzen, um $\neg Italiener$ abzuleiten, falls $\neg Trinkt\_Wein$ gegeben ist. Einige mehr implizite Konsequenzen der Kontraposition lassen sich dadurch aber immer noch nicht vermeiden. Wenn z.B.

$$\neg Trinkt\_Wein \vee \neg Trinkt\_Bier$$

gegeben ist, so können wir das Default anwenden und seine Konsequenz benutzen, um $\neg Italiener \vee \neg Trinkt\_Bier$ abzuleiten. Das ist nicht möglich, wenn die ursprüngliche Default-Regel mit Vorbedingung $Italiener$ verwendet wird.

Im Weinbeispiel waren die Konklusionen, die man erhält, zu schwach. Es gibt auch Fälle, in denen Default-Logik zu starke Ableitungen liefert. Folgendes Beispiel stammt von D. Poole [Poole, 89]

1) $true{:}Usable(x) \wedge \neg Broken(x)/Usable(x)$
2) $Broken(Left\_Arm) \vee Broken(Right\_Arm)$

Die einzige Extension enthält

$$Usable(Left\_Arm) \wedge Usable(Right\_Arm)$$

obwohl wir wissen, daß einer der Arme gebrochen ist. Reiters Definition der Extensionen garantiert nur, daß jede einzelne Konsistenzbedingung eines angewendeten Defaults mit der generierten Extension konsistent ist. Die Gesamtmenge aller Konsistenzbedingungen sämtlicher angewendeten Defaults muß nicht notwendigerweise mit der Extension konsistent sein. Deshalb erhält man in unserem Beispiel die unerwünschte Ableitung.

Makinson hat gezeigt [Makinson, 89] daß $\vdash_D$ eine weitere wichtige Eigenschaft nichtmonotoner Inferenzrelationen nicht erfüllt, die Kumulativität. Kumulativität besagt, informell, daß die Hinzunahme eines Theorems zu einer Prämissenmenge die Menge der ableitbaren Formeln nicht verändern soll. Formal:

$$\text{Wenn } X \mid\!\sim a \text{ dann } X \cup \{a\} \mid\!\sim b \text{ gdw. } X \mid\!\sim b.$$

Diese Eigenschaft scheint wesentlich für jede Inferenzrelation zu sein, wenn man Inferenz als Explizitmachen dessen versteht, was implizit in den Prämissen steckt. Denn warum sollte, wenn a implizit in X ist, das Hinzufügen von a zu X irgendeine Auswirkung haben (außer natürlich, daß man beim nächsten Mal nicht mehr so lange nach dem Beweis suchen muß).

Leider erfüllt Default-Logik, genauer $\vdash_D$, die Kumulativitätseigenschaft nicht, wie Makinsons Gegenbeispiel zeigt:

1) $true{:}p/p$
2) $p \vee q{:}\neg p/\neg p$

Diese Default-Theorie hat genau eine Extension, die p und damit $p \vee q$ enthält. Wenn jedoch $p \vee q$ als Prämisse verwendet wird, dann entsteht eine zweite Extension, die $\neg p$ enthält. Damit ist p nicht mehr skeptisch ableitbar.

Diese Schwierigkeiten haben zu einer Reihe von Modifikationen der Default Logik geführt, von denen wir hier nur einige kurz erwähnen können. Lukaszewicz [Lukaszewicz, 88] hat eine Version definiert, die auf einem zweistelligen Fixpunktoperator basiert. Das zweite Argument wird benutzt, um Konsistenzbedingungen der angewendeten Defaults mitzuführen. Ein Default wird nur angewendet, wenn seine Konsistenzbedingung nicht diejenige eines anderen angewendeten Defaults verletzt. Dadurch wird die Existenz von Extensionen garantiert, und Default-Logik wird semi-monoton, d.h., durch die Hinzunahme weiterer Defaults können neue Extensionen entstehen, existierende Extensionen werden jedoch nicht zerstört (sie können allerdings größer werden). Das Poolesche Beispiel (broken arms) läßt sich aber immer noch nicht adäquat behandeln.

In [Brewka, 91] wird CDL, eine kumulative Version der Default Logik, vorgestellt. Die Basiselemente dieser Logik sind Assertionen der Form (p, Q), wobei p eine Formel ist und Q die Menge der Konsistenzbedingungen, die man benötigt, um p abzuleiten. In CDL muß die Gesamtmenge der Konsistenzbedingungen aller angewendeten Defaults einer Extension konsistent sein. Damit läßt sich das Poolesche Beispiel wie von ihm intendiert behandeln.

Schaub und Delgrande haben eine Variante der Default Logik eingeführt, bei der die Konsistenzbedingungen der angewendeten Defaults als Kontext mitgeführt werden. Defaults werden nur dann angewendet, wenn ihre Konsistenzbedingungen in Bezug auf den gesamten Kontext (und nicht nur die ableitbaren Formeln) erfüllt sind. Auch hier führt das Poolesche Beispiel zu den erwarteten Resultaten. Diese und weitere Varianten der Default Logik sind in [Delgrande et al., 94] beschrieben.

Anstatt Reiters Logik zu modifizieren, kann man natürlich auch untersuchen, ob es Spezialfälle gibt, in denen bestimmte Probleme gar nicht erst auftreten. Dabei wird meist die Form der zulässigen Defaults eingeschränkt. In uneingeschränkten Default-Theorien kann ein Default sogar dann anwendbar sein, wenn seine Konsequenz falsch ist. Um das zu vermeiden, können wir fordern, daß die Konsequenz eines jeden Defaults von seiner Konsistenzbedingung impliziert wird (wir nehmen hier an, daß es nur eine Konsistenzbedingung gibt). Solche Defaults heißen *semi-normal* und können in folgender Form notiert werden:

$$A{:}B \wedge C/C$$

Die Beschränkung auf semi-normale Defaults hat keine großen Auswirkungen auf die Ausdrucksfähigkeit, der Nutzen von nicht semi-normalen Defaults ist sowieso eher fraglich. Andererseits gewinnt man durch diese Einschränkung auch nicht allzuviel: Existenz von Extensionen, OR und Kumulativität sind immer noch nicht erfüllt [Dix, 92], selbst wenn alle Defaults keine Vorbedingung haben, also von folgender Form sind:

$$true{:}B \wedge C/C$$

Eine interessante Klasse von Defaults, die einige wünschenswerte Eigenschaften besitzt, ist die Klasse der *normalen* Defaults. Sie haben folgende Form:

$$A{:}B/B$$

Reiter hat gezeigt [Reiter, 80], daß normale Default-Theorien immer Extensionen besitzen, außerdem sind sie semi-monoton. Allerdings ist die Ausdrucksmächtigkeit solcher Theorien nicht immer ausreichend. In der Tabelle in diesem Abschnitt haben wir bereits ein Beispiel dafür gesehen, wie man durch die geeignete Wahl von Konsistenzbedingungen Prioritäten in die Defaults hineinkodieren kann. Prioritäten spielen eine wichtige Rolle in praktischen Anwendungen, denn durch sie läßt sich die Zahl der Extensionen reduzieren.

Der Vorteil der Verwendung eines semi-normalen Defaults $d = A{:}B \wedge C/C$ für die Repräsentation von Prioritäten besteht darin, daß d blockiert wird, wenn $\neg B$ wahr ist, ohne daß man implizit annehmen muß, daß normalerweise B gilt. Deshalb wird d auch dann blockiert, wenn $\neg B$ selbst eine Default-Konklusion ist. Siehe [Brewka, 91] für eine ausführlichere Diskussion und weitere Beispiele.

Normale Default-Theorien erfüllen immer noch nicht OR und Kumulativität. Diese beiden Eigenschaften gelten erst bei einer weiteren Einschränkung, nämlich wenn nur normale Defaults ohne Vorbedingung verwendet werden [Dix, 92]. Diese Einschränkung ist allerdings erheblich, und das Resultat unterstreicht sicherlich die Bedeutung der oben erwähnten Modifikationen der Default-Logik.

2.2 Autoepistemische Logik

Moores Autoepistemische Logik (AEL) [Moore, 85] ist das derzeit sicher prominenteste Beispiel für den modalen Ansatz. Moore geht es um die Modellierung eines rationalen Agenten, der in der Lage ist, über seine eigenen Überzeugungen zu reflektieren, und der dabei über vollständige introspektive Fähigkeiten verfügt, d.h. er weiß genau, was er weiß und was er nicht weiß. Zu diesem Zweck wird ein modaler Operator L eingeführt. Lp steht für *es wird geglaubt, daß p*. Die bekannte Vogelregel wird folgendermaßen repräsentiert:

$$Vogel(x) \wedge \neg L \neg Fliegt(x) \Rightarrow Fliegt(x)$$

Es stellt sich nun die Frage, was die Mengen von Überzeugungen sind, die ein rationaler Agent auf der Basis einer gegebenen Menge von Prämissen annehmen sollte. Moore definiert diese Überzeugungsmengen, bei ihm heißen sie Expansionen einer Menge von Prämissen A, folgendermaßen:

Definition 5 *T ist eine Expansion von A genau dann, wenn*

$$T = \{p \mid A \cup Bel(T) \cup Disbel(T) \vdash p\}$$

wobei $Bel(T) = \{Lq \mid q \in T\}$, und $Disbel(T) = \{\neg Lq \mid q \notin T\}$.

Expansionen sind also Mengen von Formeln, die deduktiv abgeschlossen sind und Lp enthalten gdw. p enthalten ist, und $\neg Lp$ gdw. p nicht enthalten ist. Die Theorie

$Vogel(Tweety) \wedge \neg L \neg Fliegt(Tweety) \Rightarrow Fliegt(Tweety)$
$Vogel(Tweety)$

etwa besitzt genau eine Expansion. Da $\neg Fliegt(Tweety)$ auch unter Hinzunahme beliebiger Formeln der Form Lq oder $\neg Lq$ nicht abgeleitet werden kann, muß

$$\neg L \neg Fliegt(Tweety)$$

in der Expansion enthalten sein, und damit auch $Fliegt(Tweety)$.

Im allgemeinen Fall kann es auch mehrere Expansionen geben, nämlich immer dann, wenn, so wie im Nixon-Beispiel, unterschiedliche Defaults sich gegenseitig widersprechen.

Es konnte inzwischen gezeigt werden, daß AEL und Reiters Default Logik in sehr enger Beziehung zueinander stehen [Konolige, 88, Marek und Truszczynski, 89]. Konolige verwendet die folgende Übersetzung von Defaults in autoepistemische Formeln:

$$A{:}B_1,\ldots,B_n/C \Longrightarrow LA \wedge \neg L\neg B_1 \wedge \ldots \wedge \neg L\neg B_n \Rightarrow C$$

Da andererseits jede AEL-Formel in eine AEL-Implikation obiger Art transformiert werden kann, kann man diese Übersetzung auch in umgekehrter Richtung, also von AEL zu Default-Logik, vornehmen.

Konolige hat gezeigt, daß die Extensionen einer Default-Theorie genau dem objektiven Teil, d.h. dem Teil ohne Vorkommen des Modaloperators L, einer bestimmten Klasse von AEL-Expansionen entsprechen. Die AEL-Expansionen, für die es keine entsprechenden Default-Logik-Extensionen gibt, entstehen deshalb, weil in AEL Formeln zirkuläre Begründungen haben können. Der Wert dieser Expansionen ist deshalb auch recht fraglich. Die AEL-Theorie $\{Lp \Rightarrow p\}$, z.B., hat zwei Expansionen, von denen eine Lp und p enthält. Die entsprechende Default-Theorie dagegen besitzt nur eine Extension, in der p nicht enthalten ist.

Es sind auch nichtmonotone Systeme vorgeschlagen worden, die zwei unabhängige Modaloperatoren verwenden [Lin und Shoham, 90, Siegel, 90, Lifschitz, 91]. Die Logik MKNF von Lifschitz etwa verwendet einen epistemischen Operator K und einen Operator *not* für negation as failure (MKNF = minimal knowledge with negation as failure). In diesem System ist es möglich, zu unterscheiden zwischen Anfragen der Form „Gibt es eine Klasse, die John unterrichtet?“, formal:

$$\exists x.unterrichtet(John, x)$$

und der Anfrage „Gibt es eine (bekannte) Klasse, von der man weiß, daß John sie unterrichtet“:

$$\exists x.Kunterrichtet(John, x)$$

Diese Unterscheidung ist wichtig für Datenbanken, die disjunktive Information enthalten, und erweiterte (disjunktive) logische Programme mit echter Negation.

Ein weiterer interessanter modaler Ansatz wurde von Levesque vorgeschlagen [Levesque, 90] und von Lakemeyer weitergeführt [Lakemeyer, 92]. Levesque definiert eine monotone Modallogik, die zusätzlich zu L den Modaloperator O enthält. Op steht für „p ist alles, was gewußt wird“. Die grundlegende Idee ist die folgende: um zu bestimmen, ob beispielsweise „Tweety fliegt“ aus „Vögel fliegen typischerweise“ und „Tweety ist ein Vogel“ abgeleitet werden kann, muß überprüft werden, ob folgende Formel in der Logik gültig ist:

$$O[(Vogel(Tw) \wedge \neg L\neg Fliegt(Tw) \Rightarrow Fliegt(Tw)) \wedge Vogel(Tw)] \Rightarrow LFliegt(Tw)$$

In diesem Ansatz wird die Nichtmonotonie sozusagen vollständig in den Bereich des Operators O geschoben: $Op \Rightarrow Lq$ kann gültig sein, $O(p \wedge r) \Rightarrow Lq$ jedoch ungültig. Es stellt sich heraus, daß O eine intuitive Semantik besitzt, die auf dem Konzept möglicher Welten (possible worlds) basiert. Die AEL-Extensionen einer Formel p entsprechen genau den Interpretationen, die Op erfüllen.

2.3 Circumscription

Circumscription ist eine Technik, die es uns erlaubt, die Extension bestimmter ausgewählter Prädikate zu minimieren. Die Grundidee ist die folgende: es wird eine Präferenzrelation auf den Modellen einer Prämissenmenge definiert. Diese Präferenzrelation bevorzugt Modelle, in denen die ausgewählten Prädikate eine möglichst kleine Extension besitzen. Folgerbarkeit wird dann nicht, wie üblich, als Gültigkeit in *allen* Modellen definiert, sondern als Gültigkeit in den bezüglich dieser Präferenzrelation *besten* (minimalen) Modellen.

Syntaktisch läßt sich diese semantische Idee auf folgende Weise umsetzen: die nichtmonotonen Theoreme einer (endlichen) Prämissenmenge T werden definiert als die monotonen Theoreme von T plus gewissen zusätzlichen Formeln. Bei den zusätzlichen Formeln handelt es sich in der einfachsten Form der Circumscription um alle Instanzen eines bestimmten Formelschemas, bei anderen Varianten um eine Formel 2. Stufe. Diese Formeln dienen dazu, all die Modelle von T zu 'eliminieren', in denen die Extension des zu minimierenden Prädikates nicht minimal ist.

Eine beträchtliche Anzahl von Varianten der Circumscription sind definiert worden. Wir werden uns hier auf die einfachste Form, die Prädikaten-Circumscription, beschränken. Sie ist folgendermaßen definiert:

Definition 6 *Sei T die Konjunktion einer endlichen Menge von Prämissen, die das Prädikatensymbol P enthalten. $T(\phi)$ entstehe aus T durch Ersetzen jedes Vorkommens von P durch den Parameter ϕ. Die Prädikaten-Circumscription von P in T ist das Schema*

$$T(\phi) \wedge (\forall x.\phi(x) \Rightarrow P(x)) \Rightarrow (\forall x.P(x) \Rightarrow \phi(x)).$$

x steht hier für einen Variablenvektor entsprechend der Stelligkeit von P. Das Schema besagt intuitiv folgendes: wenn ϕ ein Prädikat ist, das alle Eigenschaften erfüllt, die in T für P festgelegt sind, so kann ϕ nicht für weniger Objekte gelten als P. Mit anderen Worten: P ist das kleinste Prädikat, das die in T für P festgelegten Eigenschaften besitzt.

Alle Instanzen dieses Schemas können zusammen mit den ursprünglichen Prämissen für Ableitungen benutzt werden. Für ϕ werden dabei Prädikatsausdrücke entsprechender Stelligkeit substituiert, das sind Lambda-Ausdrücke der Form $\lambda x_1, \ldots, x_n.F$, wobei F eine offene Formel ist und die x_i Variablen. Für diese Variablen werden bei der Substitution in F die jeweiligen Argumente von ϕ eingesetzt. Hier ist ein einfaches Beispiel [McCarthy, 80]:

$$T = Block(A) \wedge Block(B) \wedge Block(C)$$

Prädikatencircumscription von $Block$ in T ergibt das Schema:

$$\phi(A) \wedge \phi(B) \wedge \phi(C) \wedge (\forall x.\phi(x) \Rightarrow Block(x)) \Rightarrow (\forall x.Block(x) \Rightarrow \phi(x))$$

Substitution von $\lambda x.(x = A \vee x = B \vee x = C)$ für ϕ und Anwendung auf die jeweiligen Argumente ergibt:

$$\begin{array}{l}
(A = A \vee A = B \vee A = C) \wedge \\
(B = A \vee B = B \vee B = C) \wedge \\
(C = A \vee C = B \vee C = C) \wedge \\
(\forall x.(x = A \vee x = B \vee x = C) \Rightarrow Block(x)) \\
\Rightarrow \\
(\forall x.Block(x) \Rightarrow (x = A \vee x = B \vee x = C))
\end{array}$$

Die Vorbedingung dieser Implikation ist gültig und wir können ableiten, daß A, B und C die einzigen existierenden Blöcke sind. Es ist nicht schwer zu sehen, wie das Schema sich verändert, wenn die ursprüngliche Menge von Prämissen sich ändert (wenn wir etwa $Block(D)$ hinzufügen). Dadurch wird eine andere Substitution erforderlich, um die Vorbedingung der Implikation wahr zu machen und damit eine explizite Definition von $Block$ abzuleiten. Auf diese Weise entsteht die Nichtmonotonie der Circumscription.

Die Präferenzrelation $<_P$ auf Modellen, die diesem Circumscriptionsschema entspricht, ist die folgende:

Definition 7 *Sei T eine Formel, M_1 und M_2 jeweils ein Modell von T. $M_1 <_P M_2$ gdw.*

1. *M_1 und M_2 haben denselben Individuenbereich (domain),*

2. *die Extension aller Prädikate außer P ist in M_1 dieselbe wie in M_2,*

3. die Extension von P in M_1 ist eine echte Teilmenge der Extension von P in M_2.

Wie anfangs erwähnt können wir nun Folgerbarkeit als Gültigkeit in allen bezüglich $<_P$ minimalen Modellen definieren. McCarthy hat gezeigt, daß sein Schema für diesen Folgerbarkeitsbegriff korrekt ist. Für Vollständigkeit ist es erforderlich, statt des Schemas eine entsprechende Formel der Prädikatenlogik zweiter Stufe zu verwenden, in der über Prädikate quantifiziert werden kann.

McCarthy schlägt als allgemeines Prinzip für Default-Schließen mit Circumscription die Verwendung sogenannter Abnormalitätsprädikate vor. Default-Regeln wie unser Vogelbeispiel werden dabei folgendermaßen repräsentiert:

$$\forall x. Vogel(x) \wedge \neg Ab_1(x) \Rightarrow Fliegt(x)$$

Die intuitive Bedeutung der Regel ist: Vögel, die nicht abnormal sind, fliegen. Natürlich können Objekte in mancher Hinsicht normal, in anderer abnormal sein. Deshalb wird für jedes Default ein eigenes Abnormalitätsprädikat benötigt (daher der Index von Ab_1). Circumscription wird benutzt, um die Extension der Ab-Prädikate zu minimieren.

Leider hat sich herausgestellt, daß komplexere Versionen der Circumscription als die vorgestellte nötig sind, um adäquate Resultate zu erzielen. So müssen etwa bestimmte Prädikate bei der Minimierung variieren können. Des weiteren ist die Version 2. Stufe, bei der das Schema durch eine Formel zweiter Stufe ersetzt wird, aus theoretischen Gründen vorzuziehen. Einen nicht mehr ganz aktuellen, aber doch immer noch lesenswerten Überblick geben [Genesereth und Nilsson, 87].

3 Default-Schließen als Behandlung von Inkonsistenz

Wie wir in Abschnitt 2 gesehen haben, gehen die 'Standardansätze' zur Formalisierung nichtmonotonen Schließens von einer konsistenten Menge von Prämissen aus (andernfalls erhält man kein sinnvolles Resultat) und erweitern die Inferenzrelation, so daß mehr als nur die klassischen Theoreme ableitbar werden. Der im folgenden dargestellte Ansatz basiert auf einer grundsätzlich anderen Sichtweise. Was macht Default-Wissen zu Default-Wissen? Was unterscheidet es von sicheren Fakten? Sicherlich die Art und Weise, wie wir es im Falle von auftretenden Konflikten, d.h. Inkonsistenzen, behandeln. Wenn wir diese Sichtweise ernst nehmen, dann liegt die Idee nahe, Default-Schließen als einen speziellen Fall von Inkonsistenzbehandlung aufzufassen. Inkonsistente Prämissen stellen kein Problem dar, wenn wir Möglichkeiten vorsehen, mit Inkonsistenzen auf adäquate Weise umzugehen. Eine Voraussetzung dazu ist, die Inferenzrelationen so zu modifizieren, daß im Falle einer Inkonsistenz weniger, d.h. nicht alle, Formeln ableitbar werden.

In diesem Abschnitt werden wir zunächst einen allgemeinen Rahmen zur Definition nichtmonotoner Systeme vorstellen, der auf der oben diskutierten Sichtweise beruht. Wir werden dann drei konkrete Instanziierungen dieses Rahmens untersuchen. Zuerst zeigen wir, daß Pooles Ansatz zum Default-Schließen [Poole, 88] eine einfache Instanz unseres Rahmens ist. Allerdings gibt es in seinem Ansatz gewisse Beschränkungen, die darauf beruhen, daß Prioritäten zwischen Defaults nicht adäquat ausgedrückt werden können. Abschnitt 3.3 stellt eine Generalisierung von Pooles System vor, bei der verschiedene Stufen von möglichen Hypothesen eingeführt werden, die unterschiedliche Grade der Zuverlässigkeit dieser Hypothesen repräsentieren. Eine Formel ist beweisbar aus einer Prämissenmenge, wenn es möglich ist, ein konsistentes Argument für sie aus den zuverlässigsten verfügbaren Prämissen zu konstruieren.

3.1 Ein Rahmen für nichtmonotone Systeme

Eine übliche Methode zur Behandlung inkonsistenter Prämissenmengen betrachtet maximal konsistente Teilmengen der Prämissen. Da es im Allgemeinen mehr als eine solche maximal konsistente Teilmenge geben kann, wird Beweisbarkeit definiert als Ableitbarkeit aus all diesen Mengen. Die Idee hinter der Maximalitätsforderung ist klar: die verfügbare Information soll so wenig wie möglich modifiziert werden. Das Konzept der maximal konsistenten Teilmenge für sich genommen erlaubt allerdings noch nicht auszudrücken, daß z.B. *Tweety fliegt* aufgegeben werden soll und nicht *Tweety ist ein Pinguin,* wenn wir wissen, daß Pinguine nicht fliegen.

Um solche Präferenzen ausdrücken zu können, dürfen wir nicht *alle* maximal konsistenten Teilmengen betrachten, sondern nur bestimmte, die *bevorzugten* maximal konsistenten Teilmengen, oder einfacher: die *bevorzugten Teiltheorien* (ähnlich wie bei Circumscription Folgerbarkeit nicht als Gültigkeit in allen Modellen, sondern als Gültigkeit in bestimmten, bevorzugten Modellen definiert wird).

Der Begriff der bevorzugten Teiltheorie geht zurück auf [Rescher, 64]. Rescher hat eine spezielle Ordnung der maximal konsistenten Teilmengen für hypothetisches Schließen eingeführt, diese Idee jedoch nicht auf Default-Schließen angewendet.

Wir sind jetzt in der Lage, einen schwachen und einen strengen Ableitbarkeitsbegriff zu definieren:

Definition 8 *Eine Formel p ist schwach beweisbar aus einer Prämissenmenge T gdw. es eine bevorzugte Teiltheorie S von T gibt, so daß $S \vdash p$.*

Definition 9 *Eine Formel p ist streng beweisbar aus einer Prämissenmenge T gdw. für alle bevorzugten Teiltheorien S von T gilt, daß $S \vdash p$.*

„$\vdash$“ bezeichnet hier wieder die klassische Ableitbarkeitsrelation. Schwache und strenge Beweisbarkeit entsprechen, grob gesprochen, dem Enthaltensein in mindestens einer bzw. allen Extensionen in den Fixpunktansätzen zum Default-Schließen, wie etwa Default-Logik. Wir können auch den Begriff Extension einführen, und zwar auf folgende Weise:

Definition 10 *E ist eine Extension einer Prämissenmenge T gdw. es eine bevorzugte Teiltheorie S von T gibt, so daß $E = Th(S)$.*

$Th(A)$ bezeichnet hier wie üblich die Menge $\{p \mid A \vdash p\}$. Es bleibt natürlich noch zu spezifizieren, was jeweils die bevorzugten Teiltheorien sind. Zu diesem Zweck werden wir im Rest dieses Abschnittes den Prämissen T eine gewisse Struktur auferlegen. In einem Ansatz etwa (Abschnitt 3.3) werden wir T in verschiedene Levels $T_1, \ldots, T_n$ aufspalten. Diese zusätzliche Struktur wird benutzt, um die bevorzugten Teiltheorien von T zu definieren. Der Einfachheit halber werden wir häufig auch von bevorzugten Teiltheorien dieser Strukturen sprechen und die Prämissenmenge T implizit lassen.

Einen wichtigen Aspekt wollen wir hier schon diskutieren, bevor wir auf verschiedene Möglichkeiten eingehen, die bevorzugten Teiltheorien zu definieren: die beweisbaren Formeln hängen in unserem Ansatz von der syntaktischen Form der Prämissen ab. Es ist ein entscheidender Unterschied, ob z.B. eine Prämissenmenge sowohl A wie auch B enthält, oder die äquivalente einzelne Formel $A \wedge B$. Angenommen es gibt eine bevorzugte Teiltheorie S, die mit A inkonsistent ist, aber nicht mit B. Wenn beide Formeln als Prämissen gegeben sind, dann muß B in S enthalten sein. Ist jedoch die Prämisse $A \wedge B$, dann ist das nicht der Fall.

Syntaxabhängigkeit wird von Logikern häufig als Mangel empfunden. Prinzipiell wäre es möglich, durch die Einführung einer eindeutigen Normalform für Formeln dieses Problem zu umgehen. Allerdings betrachten wir in diesem Fall die Syntaxabhängigkeit gerade nicht als Nachteil. Im Gegenteil, sie erhöht die Ausdrucksmächtigkeit des Formalismus. Wir können ausdrücken, ob zwei Formeln A und B zusammen akzeptiert oder verworfen werden sollen oder nicht. Es ist durchaus sinnvoll, zwischen solchen Situationen zu unterscheiden, in denen A und B mögliche, unabhängige Hypothesen sind, und solchen, in denen $A \wedge B$ eine Hypothese darstellt.

Ein kleines Beispiel: nehmen wir an, jemand erzählt uns, er habe Uli auf dem Kaiserplatz Eis essen sehen. Nennen wir „Uli war auf dem Kaiserplatz“ A und „Uli hat Eis gegessen“ B. Erhalten wir später die zuverlässigere, mit A inkonsistente Information C: „Uli war in der Uni“, so macht es durchaus Sinn, auch B aufzugeben, obwohl B nicht mit C inkonsistent ist. In unserem Ansatz erreicht man das gerade dadurch, daß $A \wedge B$ als eine einzelne Hypothese aufgefaßt wird.

Ersetzt man eine einzelne Prämisse P durch eine logisch äquivalente einzelne Prämisse Q, so hat das keine Auswirkung auf die erzeugten Extensionen.

Wir haben bisher einen allgemeinen Rahmen für nichtmonotone Systeme definiert. Um eine spezifische Instanz, also ein konkretes nichtmonotones System, zu erhalten, muß noch definiert werden, was die bevorzugten Teiltheorien sind. Wir werden zunächst zeigen, daß Pooles Ansatz als solch eine Instanz aufgefaßt werden kann.

3.2 Pooles System

David Poole [Poole, 88] hat einen einfachen, eleganten und doch recht ausdrucksmächtigen Ansatz zum Default-Schließen entwickelt. In Pooles System definiert der Benutzer

1. eine konsistente Menge F von (geschlossenen) Formeln[1], die Fakten,
2. eine Menge Δ von, möglicherweise offenen, Formeln, die möglichen Hypothesen.

Definition 11 *Ein Szenario von F und Δ ist eine Menge $D \cup F$, wobei D eine Menge von Grundinstanzen von Formeln aus Δ ist, so daß $D \cup F$ konsistent ist.*

Definition 12 *Eine Formel g ist erklärbar aus F und Δ gdw. es ein Szenario von F und Δ gibt, in dem g abgeleitet werden kann.*

Definition 13 *Eine Extension von F und Δ ist die Menge der logischen Konsequenzen eines maximalen Szenarios von F und Δ.*

Pooles Terminologie spiegelt wieder, daß dieser Ansatz nicht nur für Vorhersagen, sondern auch für die Modellierung von Erklärungen von beobachteten Fakten verwendet werden kann. Da unser Hauptinteresse in diesem Papier auf Default-Schließen gerichtet ist, werden wir Δ auch einfach die Menge der Defaults nennen.

Es ist nicht schwierig zu sehen, daß Pooles Ansatz eine spezielle Instanz unseres Ansatzes der bevorzugten Teiltheorien darstellt. Wenn wir die bevorzugten Teiltheorien von $\Delta' \cup F$ (Δ' ergibt sich aus Δ durch Ersetzen der offenen Formeln durch alle ihre Grundinstanzen) gerade als diejenigen Teiltheorien definieren, die F enthalten, dann fallen – unter der Voraussetzung der Konsistenz von F – schwache Beweisbarkeit und Pooles Erklärbarkeit zusammen.

Hier ein kleines Beispiel, diesmal nicht aus dem Bereich der Ornithologie, das gleichzeitig eine Schwierigkeit mit Pooles Ansatz verdeutlicht. Δ besteht aus den offenen Formeln:

(1) $verletzt(x, y) \Rightarrow schuldig(x)$
(2) $greift\_an(x, y) \Rightarrow notwehr(y)$
(3) $notwehr(x) \Rightarrow \neg schuldig(x)$

F besteht aus folgenden Fakten:

(4) $greift\_an(Hans, Peter)$
(5) $verletzt(Peter, Hans)$

Jetzt ist erklärbar $\neg schuldig(Peter)$, aber auch $schuldig(Peter)$ und damit ebenso $\neg notwehr(Peter)$. Der Grund ist, daß wir (1) verwenden können, um (2) und (3) zu blockieren. Was in obigem Beispiel fehlt, ist die Festlegung von geeigneten Prioritäten zwischen den Defaults.

Poole hat gezeigt, wie einige dieser Probleme in seinem System gelöst werden können. Er benutzt dabei benannte Defaults: für eine Hypothese $w(x) \in \Delta$ mit freien Variablen x wird ein neues

[1] Eine Formel ist geschlossen, wenn sie keine ungebundenen Variablen enthält, sonst offen.

Prädikatensymbol p_w derselben Stelligkeit eingeführt. Poole zeigt, daß $w(x)$ äquivalent durch $p_w(x)$ ersetzt werden kann, wenn die Formel

$$\forall x.p_w(x) \Rightarrow w(x)$$

zu F hinzugefügt wird. Poole benutzt dafür die Notation $p_w(x) : w(x)$ als abkürzende Schreibweise. Das zeigt, daß die Mächtigkeit des Systems nicht beeinträchtigt wird, wenn man nur atomare Formeln in Δ zuläßt.

Die Benutzung von Namen erlaubt es, die Anwendbarkeit von Defaults bei Bedarf zu blockieren. Wenn ein Default $p_w(x)$ in einer Situation s nicht anwendbar sein soll, so können wir einfach $\forall x.s \Rightarrow \neg p_w(x)$ zu den Fakten hinzufügen. [Poole, 88] enthält zahlreiche überzeugende Beispiele dafür, wie diese Technik verwendet werden kann. Dabei ist der Unterschied zwischen allquantifizierten Formeln und Schemata mit freien Variablen wesentlich: eine allquantifizierte Formel in Δ, etwa $\forall x.P(x) \Rightarrow Q(x)$, wird durch ein Gegenbeispiel in F, etwa $P(a)$ und $\neg Q(a)$, außer Kraft gesetzt. Enthält dagegen Δ die offene Formel $P(x) \Rightarrow Q(x)$, die alle ihre Grundinstanzen repräsentiert, so wird durch das Gegenbeispiel nur die entsprechende Instanz außer Kraft gesetzt.

Unser obiges Beispiel kann mit dieser Technik etwa folgendermaßen dargestellt werden: Δ wird zu der Menge

$$\{d_1(x,y), d_2(x,y), d_3(x)\}$$

und aus F wird

(1) $\forall x,y.d_1(x,y) \Rightarrow (verletzt(x,y) \Rightarrow schuldig(x))$
(2) $\forall x,y.d_2(x,y) \Rightarrow (greift\_an(x,y) \Rightarrow notwehr(y))$
(3) $\forall x,y.d_3(x) \Rightarrow (notwehr(x) \Rightarrow \neg schuldig(x))$
(4) $greift\_an(Hans, Peter)$
(5) $verletzt(Peter, Hans)$
(6) $\forall x,y.greift\_an(x,y) \Rightarrow \neg d_1(y,x)$

Wichtig ist hier insbesondere (6), weil diese Formel explizit die Anwendung von (1) in bestimmten Fällen blockiert. Jetzt ist in unserem Beispiel, wie gewünscht, $schuldig(Peter)$ nicht mehr erklärbar.

Poole hat gezeigt, daß mit dieser Technik sehr viele der Standardbeispiele zum Default-Schließen aus der Literatur adäquat behandelt werden können [Poole, 88]. Allerdings ist diese Art, Prioritäten zwischen Defaults auszudrücken, in einigen Fällen äußerst mühselig und unpraktisch [Brewka, 90]. Dieser Mangel ist die Motivation für die Verallgemeinerungen, die wir in den nächsten Abschnitten beschreiben werden.

Bleibt noch zu sagen, daß mit dem System THEORIST eine recht effiziente Prolog-Implementierung von Pooles Ansatz zur Verfügung steht [Poole et al., 86].

3.3 Zuverlässigkeitsstufen

Die Grundidee des Pooleschen Ansatzes war es, zwischen genau zwei Typen von Prämissen zu unterscheiden: solchen, die mit Sicherheit gelten (und konsistent sein) müssen (Fakten), und solchen, die weniger zuverlässig sind und möglicherweise nicht gelten (Hypothesen).

Wir verallgemeinern diese Idee in zwei Hinsichten: zum einen fordern wir nicht die Konsistenz der zuverlässigsten Formeln. In unserer Verallgemeinerung ist jede Formel prinzipiell widerlegbar. Zum anderen führen wir mehr als genau zwei Typen von Prämissen ein. Die Prämissen werden in verschiedene Stufen oder Levels eingeteilt. Die Idee dabei ist, daß die verschiedenen Stufen verschiedene Grade der Zuverlässigkeit repräsentieren. Die erste Stufe enthält die zuverlässigste Information, die zweite Stufe die nächst zuverlässige, usw. Wenn Inkonsistenzen auftreten, dann wird jeweils die zuverlässigere Information bevorzugt. Intuitiv gesprochen ist eine Formel dann ableitbar, wenn ein Argument für sie aus der zuverlässigsten verfügbaren Information konstruiert werden kann.

Natürlich kann es dabei widersprüchliche Information derselben Zuverlässigkeit geben. In diesem Fall erhalten wir wieder multiple Extensionen, ähnlich wie in Reiters Default Logik. Die Eigenschaft, daß es keine prinzipiell unwiderlegbaren Prämissen gibt, macht es möglich, alle Stufen auf uniforme Weise zu behandeln. Z.B. können wir stets Information hinzufügen, die zuverlässiger ist als die gegenwärtig erste Stufe.

Diese intuitiven Ideen müssen natürlich im Rahmen unseres Ansatzes präzisiert werden. Das heißt, wir müssen definieren, was die bevorzugten Teiltheorien in diesem Fall sein sollen.

Definition 14 *Eine Default-Theorie T ist ein Tupel $(T_1, \ldots, T_n)$, wobei jedes T_i eine Menge klassischer Formeln erster Ordnung ist.*

Information in T_i ist zuverlässiger als solche in T_j, wenn $i < j$. Ein Default wie *Vögel fliegen* kann wiederum repräsentiert werden als die Menge aller Grundinstanzen des Schemas $Vogel(x) \Rightarrow Fliegt(x)$. Der Einfachheit halber schreiben wir $T_i = \{\ldots, P(x), \ldots\}$ um auszudrücken, daß T_i alle Grundinstanzen von $P(x)$ enthält.

Definition 15 *Sei $T = (T_i, \ldots, T_n)$ eine Default-Theorie. $S = S_1 \cup \ldots \cup S_n$ ist eine bevorzugte Teiltheorie von T gdw. für alle k $(1 \leq k \leq n)$ gilt: $S_1 \cup \ldots \cup S_k$ ist eine maximal konsistente Teilmenge von $T_1 \cup \ldots \cup T_k$.*

In anderen Worten: um zu einer bevorzugten Teiltheorie zu gelangen, nimmt man eine beliebige maximal konsistente Teilmenge von T_1, fügt so viele Formeln von T_2 hinzu, wie das konsistent möglich ist, usw. bis man bei T_n angelangt ist.

Die folgenden einfachen Beispiele zeigen, wie die verschiedenen Stufen benutzt werden können, um Prioritäten zwischen Defaults auszudrücken:

$$T1 : \{Vogel(Tweety), \forall x.Pinguin(x) \Rightarrow \neg Fliegt(x)\}$$
$$T2 : \{Vogel(x) \Rightarrow Fliegt(x)\}$$

$Fliegt(Tweety)$ ist streng beweisbar.

$$T1 : \{Vogel(Tweety), \forall x.Pinguin(x) \Rightarrow \neg Fliegt(x), Pinguin(Tweety)\}$$
$$T2 : \{Vogel(x) \Rightarrow Fliegt(x)\}$$

Jetzt ist $\neg Fliegt(Tweety)$ streng beweisbar. Das Beispiel illustriert auch noch einmal die wichtige Unterscheidung zwischen Schemata und allquantifizierten Formeln. Würde in $T2$ die entsprechende quantifizierte Formel benutzt, so würde ein einziger nichtfliegender Vogel genügen, um die Regel außer Kraft zu setzen.

Gibt es auch fliegende Pinguine, dann können wir folgende Repräsentation benutzen, die der spezielleren Pinguin-Regel höhere Priorität verleiht als der Vogel-Regel:

$$\begin{aligned} T1 : \{&Vogel(Tweety), Pinguin(Tweety), Pinguin(Tim), \\ &Fliegt(Tim), Vogel(Jim)\} \end{aligned}$$
$$T2 : \{Pinguin(x) \Rightarrow \neg Fliegt(x)\}$$
$$T3 : \{Vogel(x) \Rightarrow Fliegt(x)\}$$

Jetzt fliegen (streng beweisbar) Tim und Jim, $Tweety$ fliegt nicht.

Hier ist eine Repräsentation unseres Notwehr-Beispiels ohne benannte Defaults.

$$T1 : \{greift\_an(Hans, Peter), verletzt(Peter, Hans)\}$$
$$T2 : \{greift\_an(x, y) \Rightarrow notwehr(y), notwehr(x) \Rightarrow \neg schuldig(x)\}$$
$$T3 : \{verletzt(x, y) \Rightarrow schuldig(x)$$

Wie gewünscht ist $\neg schuldig(Peter)$ streng beweisbar.

Bisher haben wir in unseren Beispielen die Formeln selbst in die entsprechenden Stufen einsortiert. Es sind auch Verfahren entwickelt worden, die automatisch für eine vorgegebene Menge von Default-Regeln eine Einsortierung vornehmen und dabei die Spezifizität der jeweiligen Regeln berücksichtigen [Pearl, 90, Kraus et al., 90].

Eine Verfeinerung der hier vorgestellten Behandlung von Zuverlässigkeitsstufen wurde von Benferhat und seinen Toulouser Kollegen [Benferhat et al., 93] sowie von Lehmann [Lehmann, 92] vorgeschlagen. Dabei wird bei der Definition bevorzugter Teiltheorien zusätzlich die Anzahl der pro Stufe akzeptierten Default-Regeln berücksichtigt. Interessanterweise erfüllt die dadurch definierte strenge Beweisbarkeitsrelation die Eigenschaft der rationalen Monotonie, die besagt, daß Konklusionen erhalten bleiben, solange die Prämissen nicht um unerwartete Formeln erweitert werden. Formal:

$$\text{Wenn } X \mid\sim a \text{ und nicht } X \mid\sim \neg b \text{ dann } X \cup \{b\} \mid\sim a.$$

Manchmal möchte man offen lassen, ob eine Formel p größere, kleinere oder dieselbe Zuverlässigkeit besitzt wie eine andere Formel q. In diesem Fall kann man anstelle der Zuverlässigkeitsstufen auch eine beliebige strikte partielle Ordnung der Prämissen zulassen, die wiederum die relative Zuverlässigkeit der Formeln ausdrückt. Eine Default-Theorie ist dann ein Paar $(T, <)$, wobei die Relation $<$ definiert ist über T und $p < q$ ausdrückt, daß p zuverlässiger ist als q. Eine bevorzugte Teiltheorie erhält man, indem man die Prämissen in einer beliebigen mit $<$ kompatiblen Reihenfolge, beginnend mit den relevantesten, testet und jeweils die Prämissen mit aufnimmt, die konsistent mit den bereits akzeptierten Prämissen sind.

Insgesamt scheint es ein Vorteil des in diesem Abschnitt beschriebenen Ansatzes zu sein, daß er näher an der klassischen Logik ist als viele andere Systeme: wir brauchen keine Modaloperatoren, keine nichtklassischen Inferenzregeln, keine Fixpunktkonstruktionen, keine Logik höherer Ordnung oder Abnormalitätsprädikate. Des weiteren ist das Problem der Behandlung inkonsistenter Information – ein Problem, das jede Modellierung intelligenten Verhaltens sowieso behandeln muß – implizit gelöst.

Andererseits ist der hier vorgestellte Ansatz natürlich in bestimmter Hinsicht weniger ausdrucksfähig als etwa Reiters Default Logik: Defaults in diesem Ansatz entsprechen, wie Poole gezeigt hat, normalen Reiter-Defaults ohne Vorbedingungen. Deshalb wurden Methoden der Behandlung von expliziten Präferenzen zwischen Defaults auch für die Default Logik entwickelt, z.B. in [Brewka, 94]. Diese Arbeit geht noch einen Schritt weiter und zeigt, wie man die Information über die Präferenzen innerhalb der logischen Sprache selbst ausdrücken kann. Damit werden auch die Präferenzen selbst dynamisch ableitbar und Strategien zur Lösung von Konflikten zwischen Defaults können deklarativ beschrieben werden. Das ist für viele Anwendungen äußerst hilfreich, wie das zitierte Papier anhand eines Beispiels aus dem juristischen Schließen illustriert.

4 Nichtmonotonie und Logikprogrammierung

In diesem Abschnitt wollen wir die Zusammenhänge zwischen nichtmonotonen Logiken und Logikprogrammierung darstellen. Es stellt sich heraus, daß die Standardsemantiken der Logikprogrammierung in enger Beziehung insbesondere zu Reiters Default Logik stehen. Somit lassen sich Systeme aus der Logikprogrammierung, sofern sie diese Semantiken tatsächlich implementieren, als eingeschränkte Beweissysteme für die Default Logik auffassen.

Tatsächlich sind Einschränkungen bei der Implementierung von Beweissystemen für nichtmonotone Logiken unumgänglich, zumindest wenn es sich um nichtmonotone Logiken erster Stufe handelt. Diese sind in ihrer allgemeinsten Form nämlich nicht semi-entscheidbar, d.h. es existiert nachweislich kein korrektes und vollständiges Beweisverfahren, das immer terminiert. Wir müssen also mit bestimmten Einschränkungen leben bzw. auf Vollständigkeit unserer Beweisverfahren verzichten.

Wie bereits in der Einleitung erwähnt, führt die besondere Behandlung der Negation (negation as failure) in der Logikprogrammierung zu Nichtmonotonie. Ursprünglich wurde die Bedeutung dieser Art von Negation prozedural erklärt: wenn Beweiser ein Literal der Form $notL$ abzuleiten haben, so wird ein Beweis für L gestartet. Gelingt dieser Beweis, so gilt $notL$ als falsch. Falls der Beweis nicht gelingt und der Beweiser terminiert, so gilt $notL$ als wahr.

Diese Art der Charakterisierung der Bedeutung von Logikprogrammen ist aus zwei Gründen unbefriedigend: 1) die Semantik eines Programmes hängt von einer zugrundeliegenden Beweisprozedur ab (eigentlich sollte eine Semantik ja Maßstab für Korrektheit und Vollständigkeit einer Beweisprozedur sein), 2) die Bedeutung von Programmen, für die der Beweiser nicht terminiert, bleibt offen.

Es sind deshalb eine ganze Reihe von Semantiken für Logikprogramme entwickelt worden, die unabhängig von der zugrundeliegenden Beweisprozedur sind. Die zwei wichtigsten Semantiken, nämlich die Semantik stabiler Modelle und die wohlfundierte Semantik, wollen wir hier vorstellen.

4.1 Stabile Modelle

Zunächst müssen wir präzisieren, was unter einem Logikprogramm zu verstehen ist. Im einfachsten Fall bestehen solche Programme aus Regeln der Form

$$c \leftarrow a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m$$

wobei a_i, b_j und c atomare Formeln sind. Wenn in den Regeln Variablen auftauchen, so repräsentieren die Regeln alle ihre Grundinstanzen. Verallgemeinerungen, die Disjunktionen in den Regelköpfen oder zwei Arten von Negation zulassen, werden später kurz diskutiert.

Ein stabiles Modell M ist nun, intuitiv, eine Menge von Grundatomen, die man aus den Regeln eines Programmes ableiten kann, wenn man die Negation in den Regelkörpern in Bezug auf M selbst auswertet. Es begegnet uns also wieder die Selbstbezüglichkeit, die uns schon von Reiters Default Logik bekannt ist. Hier ist die formale Definition:

Definition 16 *Sei P ein Logikprogramm, M eine Menge von Atomen. Die M-Reduktion von P, P^M, ist das Programm, das man aus P erhält, indem man*

1. *alle Regeln entfernt, in deren Körper $not\ l$ für ein $l \in M$ vorkommt,*
2. *alle negierten Atome aus den übrigen Regeln entfernt.*

M heißt stabiles Modell von P, wenn M die kleinste unter den Regeln in P^M abgeschlossene Menge von Atomen ist.

Die in dieser Definition durchgeführte Reduktion des Programms entspricht gerade der Auswertung von *not* in Bezug auf M. Bezeichnen wir die kleinste Menge von Atomen, die unter einer Menge R von Regeln ohne Negation abgeschlossen[2] ist, mit $Cn(R)$, so können wir die stabilen Modelle von P auch als Fixpunkte eines Operators γ_P einführen, der folgendermaßen definiert ist:

$$\gamma_P(X) = Cn(P^X)$$

M ist also stabiles Modell von P gdw. $M = \gamma_P(M)$. Ein Atom wird als ableitbar betrachtet, wenn es in allen stabilen Modellen eines Programms enthalten ist.

Hier ein kleines Beispiel:

[2] S ist abgeschlossen unter einer Regelmenge R genau dann wenn die Konsequenzen aller Regeln, deren Vorbedingungen in S sind, bereits in S enthalten sind.

$a \leftarrow d, not\ b$
$b \leftarrow d, not\ a$
$c \leftarrow a$
$c \leftarrow b$
$d \leftarrow not\ e$

Es existieren zwei stabile Modelle, $M_1 = \{a, c, d\}$ sowie $M_2 = \{b, c, d\}$. Wir zeigen, daß M_1 stabiles Modell ist. Dazu bilden wir P^{M_1}. Da $a \in M_1$, entfällt die zweite Regel. Die negierten Literale der übrigen Regeln entfallen ebenso, und wir erhalten:

$a \leftarrow d$
$c \leftarrow a$
$c \leftarrow b$
d

Die kleinste unter diesen Regeln abgeschlossene Menge von Atomen ist M_1, und damit ist M_1 stabiles Modell. Der Nachweis für M_2 ist analog. Da c in allen stabilen Modellen enthalten ist, ist c ableitbar.

Was hat das alles Reiters Default Logik zu tun? Die Antwort ist einfach: es handelt sich um einen Spezialfall. Die Regel

$$c \leftarrow a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m$$

entspricht dem Default

$$a_1 \wedge \ldots \wedge a_n : \neg b_1, \ldots, \neg b_m / c$$

Wir können also jedes Logikprogramm P in eine Default Theorie $Tr(P)$ übersetzen, indem wir die einzelnen Regeln in Defaults übersetzen. Die Menge W bleibt dabei leer. Es gilt nun folgender Zusammenhang:

Proposition 1 *Sei P ein Logikprogramm, $Tr(P)$ seine Übersetzung in Default Logik. M ist stabiles Modell von P gdw. es eine Extension E von $Tr(P)$ gibt, so daß $E \cap Atoms = M$.*

Hierbei bezeichnet *Atoms* die Menge der Atome. Die Definition der stabilen Modelle kann also als Spezialisierung der Reiterschen Definition der Extensionen aufgefaßt werden, zugeschnitten auf die einfachere Sprache der Logikprogramme.

4.2 Wohlfundierte Semantik

Die ursprüngliche Formulierung der wohlfundierten Semantik (wellfounded semantics) von Gelder, Ross and Schlipf [Gelder et al., 91] basiert auf einem bestimmten partiellen Modell. Przymusinski rekonstruierte diese Definition in 3-wertiger Logik [Przymusinski, 90]. Wir verwenden hier die Darstellung von Baral and Subrahmanian [Baral und Subrahmanian, 91], die auf dem bereits bekannten Operator γ_P beruht, ihn aber völlig anders benutzt.

Es ist leicht zu sehen, daß dieser Operator anti-monoton ist, d.h. $S \subseteq S'$ impliziert $\gamma_P(S') \subseteq \gamma_P(S')$. Das liegt daran, daß S' bei der Programmreduktion mindestens soviele Regeln aus P eliminieren kann wie S. Die zweifache Hintereinanderausführung von γ_P definiert damit einen monotonen Operator. Nach dem berhmten Knaster-Tarski Theorem [Tarski, 55] besitzt jeder monotone Operator einen kleinsten Fixpunkt. Dieser definiert nun gerade die Menge der ableitbaren Atome.

Definition 17 *Sei P ein Logikprogramm, A ein Atom. A ist wohlfundierte Konklusion von P gdw. A im kleinsten Fixpunkt von ${\gamma_P}^2$ enthalten ist.*

Der kleinste Fixpunkt von ${\gamma_P}^2$ kann iterativ erzeugt werden, indem man ausgehend von der leeren Menge ${\gamma_P}^2$ immer wieder anwendet, bis keine neuen Atome mehr hinzukommen. Diese Vorgehensweise verdeutlicht auch die eigentliche Intuition hinter dieser Definition. Nehmen wir an, wir wissen bereits, daß eine Menge von Atomen S ableitbar ist. $\gamma_P(S)$ ist dann die Menge aller Atome, die aufgrund dieser Information überhaupt noch als mögliche Konklusionen in Frage kommen. Wenden wir nun γ_P auf diese Menge an, so können wir sicher sein, daß es sich bei den nun resultierenden Atomen wieder um ableitbare Atome handelt.

Wir wollen das anhand des Beispiel-Programmes aus dem letzten Abschnitt illustrieren, das wir der Lesbarkeit halber wiederholen wollen:

$$
\begin{aligned}
&a \leftarrow d, not\ b \\
&b \leftarrow d, not\ a \\
&c \leftarrow a \\
&c \leftarrow b \\
&d \leftarrow not\ e
\end{aligned}
$$

Wir erhalten folgende Mengen von Atomen:

$\gamma_P(\emptyset) = \{a, b, c, d\}$	keine Regeln werden eliminiert
${\gamma_P}^2(\emptyset) = \{d\}$	die ersten beiden Regeln werden eliminiert
$\gamma_P(\{d\}) = \{a, b, c, d\}$	keine Regeln werden eliminiert
${\gamma_P}^2(\{d\}) = \{d\}$	kleinster Fixpunkt erreicht

Damit ist d die einzige wohlfundierte Konklusion unseres Programmes. a, b und c sind zwar potentielle Konklusionen, können aber nicht abgeleitet werden.

Es ist kein Zufall, daß die wohlfundierten Konklusionen hier eine Teilmenge der Atome sind, die unter der stabilen Semantik abgeleitet werden können. Das ist in der Tat immer so. Die wohlfundierte Semantik ist also korrekt bzgl. der stabilen Semantik (und damit wegen des Zusammenhangs stabile Semantik/Default Logik auch korrekt bzgl. Default Logik), und wir können sie als Approximation der letzteren Semantik auffassen. Tatsächlich ist das aussagenlogische Ableitbarkeitsproblem (ist ein Atom A aus einem aussagenlogischen Programm P ableitbar?) der wohlfundierten Semantik in quadratischer Zeit lösbar, während es für die stabile Semantik co-NP-vollständig ist. Dieser beachtliche Komplexitätsgewinn hat viel zur Popularität der wohlfundierten Semantik beigetragen.

4.3 Erweiterungen und Inferenzsysteme

Die syntaktischen Beschränkungen der Logikprogramme, die wir bisher behandelt haben, sind erheblich. Deshalb hat man immer wieder versucht, die Syntax der Regeln zu verallgemeinern. Gegenstand der Forschung waren insbesondere Logikprogramme mit zwei Arten von Negation, in denen neben dem *not* auch das klassische Negationssymbol $\neg$ auftreten kann, sowie Regeln mit Disjunktion in den Köpfen.

Die Einführung einer zweiten, "echten Negation ist relativ unproblematisch. Gelfond und Lifschitz [Gelfond und Lifschitz, 90] haben gezeigt, wie sich stabile Modelle zu sogenannten Antwortmengen verallgemeinern lassen, die zwei Arten von Negation adäquat behandeln. Entsprechende Erweiterungen der wohlfundierten Semantik wurden von verschiedenen Autoren vorgeschlagen [Przymusinski, 91, Pereira und Alferes, 89, Baral und Gelfond, 94, Lifschitz, 95, Brewka, 96].

Mehr Schwierigkeiten bereiten disjunktive Logikprogramme. Hier gibt es eine ganze Reihe von konkurrierenden Ansätzen, stabile und wohlfundierte Semantik entsprechend zu erweitern. Eine gute Übersicht bietet [Brass und Dix, 99].

Die Logikprogrammierung hat nicht nur zur Identifikation interessanter Teilklassen nichtmonotoner Logiken beigetragen, sondern aus diesem Bereich stammen auch einige der interessantesten derzeit existierenden nichtmonotonen Inferenzsysteme:

- **XSB**, entwickelt von David Warren und seiner Gruppe in Stony Brook [Sagonas et al., 96], implementiert die wohlfundierte Semantik. Mit seiner recht komfortablen Programmierumgebung ist es das derzeit wohl am weitesten in Richtung Produkt entwickelte System. Es basiert auf einer Variante der SLD-Resolution, der sogenannten SLG-Resolution, die es ermöglicht, Schleifen zu behandeln, die in üblichen Prolog-Systemen zu Nichtterminierung führen.

- **smodels**, entwickelt von einer Gruppe um Ilkka Niemelä in Helsinki [Niemelä und Simons, 97], ist das derzeit wohl effizienteste System für Logikprogramme unter stabiler Semantik. So ist etwa die smodels-Implementierung eines Planungssystems nach Aussagen der Autoren mit den schnellsten derzeit existierenden Planungssystemen konkurrenzfähig.

- **dlv**, entwickelt von einer Gruppe an der TU Wien [Eiter et al., 98], berechnet stabile Modelle für disjunktive Logikprogramme mit zwei Arten von Negation. Das System erkennt syntaktische Teilklassen von Programmen mit geringerer Komplexität und und verwendet für entsprechende Programme (bzw. Programmteile) äußerst effiziente Algorithmen.

Alle genannten Systeme sind frei verfügbar. Regelmäßig aktualisierte Hinweise zu Bezugsmodalitäten und Weiterentwicklungen sowie Information über weitere nichtmonotone Systeme findet man im Web unter http://www.uni-koblenz.de/ag-ki/LP.

Einen sehr guten Überblick über die Systeme und eine Vielzahl weiterer Beweistechniken für nichtmonotone Logiken gibt [Dix et al., 99].

5 Ausblick

Wie wir gesehen haben, ist eine große Zahl von verschiedenen Ansätzen vorgeschlagen worden, nichtmonotone Schlußweisen zu formalisieren. Ihre Eigenschaften und ihr Verhältnis zueinander (Ausdrucksmächtigkeit, Übersetzbarkeit usw.) wurden intensiv untersucht, und der Bezug zur Logikprogrammierung wurde geklärt. Außerdem stehen eine Reihe von interessanten Implementierungen zur Verfügung.

Es bleibt die Frage nach Anwendungen. Gibt es sie berhaupt? Und wenn ja, stehen theoretischer Aufwand und Anwendungsnutzen in ausgewogenem Verhältnis? Es ist ja nicht zu bestreiten, daß die Literatur in diesem Gebiet überwiegend theoretischer Natur ist.

Die Antwort auf obige Fragen hängt unter anderem davon ab, was man eigentlich unter Anwendung versteht. Wie wir in der Einleitung gesehen haben, gibt es eine Reihe von nichtmonotonen Systemen, die in der KI und Informatik allgemein Verbreitung gefunden haben: Frame Systeme mit nichtmonotoner Vererbung, Prolog, Datenbanken mit Closed World Assumption etc. Nichtmonotone Logiken sind zum Verständnis dieser Systeme und ihres Verhaltens (zur Beschreibung der Semantik der Systeme) von erheblicher Bedeutung und können als das theoretische Fundament verstanden werden, das diese Systeme auf wissenschaftlich sicheren Boden stellt. In diesem Sinne kann man auch diese Systeme, die mehr oder weniger eingeschränkte Repräsentationssprachen zur Verfügung stellen, als "Anwendungen nichtmonotoner Logiken verstehen.

Man kann den Begriff Anwendung aber auch enger fassen im Sinne der Lösung eines konkreten Problems oder einer Problemklasse durch Repräsentation in einem nichtmonotonen Inferenzsystem. Beispiele erfolgreicher Anwendungen dieser Art sind etwa die modellbasierte Diagnose, bei der das normale Verhalten der Komponenten eines Systems beschrieben und mit dem tatsächlich beobachteten Verhalten verglichen wird. Dabei verwendet man Defaults, die die Korrektheit von Komponenten ableiten, sofern dies mit den Beobachtungen konsistent ist. Diejenigen Komponenten, für die die Korrektheitsannahmen zu Widersprüchen führen, bilden jeweils mögliche Diagnosen.

Ein weiterer Bereich, in dem nichtmonotone Logiken erfolgreich angewendet wurden, ist das juristische Schließen. So hat etwa Gordon [Gordon, 95] eine von Geffner vorgeschlagene nichtmonotone Logik (conditional entailment [Geffner, 92]) verwendet, um juristische Vorverhandlungen (pleading) zu modellieren.

Ein nach wie vor äußerst aktives Forschungsgebiet ist die Modellierung von Handlungen (reasoning about action) und ihrer Effekte. Wie bereits in der Einleitung beschrieben, war dieses Gebiet wesentlich für die Entstehung der Nichtmonotonie-Forschung. Allerdings zeigte sich gerade hier sehr schnell ein Phänomen, das auch in anderen Anwendungsgebieten häufig zu beobachten ist: die Verwendung einer nichtmonotonen Logik bzw. eines nichtmontonen Inferenzsystems ist zwar ein wichtiger Schritt zur Lösung des Problems, aber nicht selbst schon die Lösung. Hatte man ursprünglich geglaubt, das Frame-Problem und verwandte Probleme durch einfache Defaultregeln in den Griff zu bekommen, die die Persistenz von Eigenschaften gewährleisten sollten, so stellte sich das spätestens nach Publikation des berühmten Yale-Shooting-Problems [Hanks und McDermott, 87] als Illusion heraus. Heute besteht weitgehend Einigkeit darüber, daß etwa der Begriff der Kausalität modelliert werden muß, wenn Effekte von Handlungen adäquat beschrieben werden sollen. Wesentlich ist also nicht nur daß, sondern insbesondere auch wie und in welchem Kontext ein nichtmonotoner Ansatz für ein konkretes Problem verwendet wird. [3]

Als Beispiel für ein äußerst ambitioniertes Langzeitprojekt, in dem Handlungsmodellierung eine zentrale Rolle spielt, sei hier das von Erik Sandewall geleitete WITAS UAV (unmanned aerial vehicle) Projekt genannt, das von der schwedischen Wallenberg Foundation gefördert wird und die Entwicklung eines intelligenten autonomen Luftfahrzeugs zum Ziel hat. Informationen findet man unter http://www.ida.liu.se/ext/witas/.

In dieser knappen Darstellung konnten wir nur einige der relevantesten Ansätze vorstellen. Als weiterführende Literatur in diesem Gebiet wären die *Readings in Nonmonotonic Reasoning*, herausgegeben von M. Ginsberg (Morgan Kaufmann, 1987), zu nennen. Darin sind die bis zum Datum der Herausgabe wichtigsten Originalartikel enthalten. Das Buch [Brewka et al., 94] gibt einen wesentlich ausführlicheren Überblick, als es hier möglich war. Es geht auch auf den konditionalen Ansatz von Kraus, Lehmann und Magidor [Kraus et al., 90] oder den verwandten Ansatz von Pearl ein, der die Zusammenhänge zwischen nichtmonotonem Schließen und Schließen mit (infinitesimalen) Wahrscheinlichkeiten erhellt [Pearl, 90]. Weitere empfehlenswerte Bücher sind [Marek und Truszczynski, 93], [Schaub, 98] sowie [Antoniou, 97]. Eine exzellente Übersicht über verschiedene Eigenschaften nichtmonotoner Inferenzrelationen und die sie erfüllenden Ansätze bietet [Makinson, 94].

Literatur

[Antoniou, 97] Antoniou, Grigoris, Nonmonotonic Reasoning, MIT Press, 1997

[Baral und Gelfond, 94] Baral, C., Gelfond, M., Logic Programming and Knowledge Representation, Journal of Logic Programming, 19,20:73-148, 1994

[Baral und Subrahmanian, 91] Baral, C., Subrahmanian, V.S., Duality between Alternative Semantics of Logic Programs and Nonmonotonic Formalisms, Intl. Workshop on Logic Programming and Nonmonotonic Reasoning, 1991

[Benferhat et al., 93] Benferhat, Salem, Cayrol, Claudette, Dubois, Didier, Lang, Jerome, Prade, Henri: Inconsistency Management and Prioritized Syntax-Based Entailment, Proc. IJCAI-93, Chambery, France, 1993

[Boutilier, 94] Boutilier, Craig: Conditional Logics of Normality: A Modal Approach, Artificial Intelligence 68, No 1, July 1994

[3] Für an dem Thema Reasoning about Action and Change Interessierte bieten die von Erik Sandewall betreuten Seiten der Electronic Transactions on AI (http://www.ida.liu.se/ext/etai/rac/) ein hervorragendes Forum mit aktuellen Artikeln, Übersichtsbeiträgen und Diskussionen.

[Brass und Dix, 99] Brass, Stefan, Dix, Jürgen, Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation, Journal of Logic Programming 38(2), 111–157, 1999

[Brewka, 90] Brewka, Gerhard: Nonmonotonic Reasoning – Logical Foundations of Commonsense, Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1990

[Brewka, 91] Brewka, Gerhard: Cumulative Default Logic – In Defense of Nonmonotonic Inference Rules, Artificial Intelligence 50, 1991

[Brewka, 94] Brewka, Gerhard: Reasoning About Priorities in Default Logic, Proc. AAAI 94, Seattle, 1994

[Brewka, 96] G. Brewka. Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences. *Journal of Artificial Intelligence Research*, 4:19–36, 1996.

[Brewka et al., 94] Brewka, Gerhard, Dix, Jürgen, Konolige, Kurt: Nonmonotonic Reasoning - An Overview, CSLI Publications, Stanford, 1997

[Delgrande, 88] Delgrande, James P.: An Approach to Default Reasoning Based on a First-Order Conditional Logic: Revised Report, Artificial Intelligence 36, 1988

[Delgrande et al., 94] Delgrande, James P., Schaub, Torsten, und W. Ken Jackson, Alternative Approaches to Default Logic. Artificial Intelligence 70(1-2): 167-237, 1994

[Doyle, 79] Doyle, J.: A Truth Maintenance System, Artificial Intelligence 12, 1979

[Dix, 92] Dix, Jürgen: Default Theories of Poole-Type and a Method for Constructing Cumulative Versions of Default Logic, Proc. ECAI 92, Wien, 1992

[Dix et al., 99] Dix, Jürgen, Furbach, Ulrich, Niemelä, Ilkka, Nonmonotonic Reasoning: Towards Efficient Calculi and Implementations, to appear in: A. Robinson, A. Voronkov, Handbook of Automated Reasoning, Elsevier

[Eiter et al., 98] Eiter, T., Leone, N., Mateis, C., Pfeifer, G., Scarcello, F.: The KR System dlv: Progress Report, Comparisons and Benchmarks, Proc. 6th Intl. Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 1998

[Gabbay, 85] Gabbay, Dov: Theoretical Foundations for Non-monotonic Reasoning in Expert Systems, In: Logics and Models of Concurrent Systems, Berlin, Springer Verlag, 1985

[Geffner, 92] Geffner, Hector, Default Reasoning: Causal and Conditional Theories, Cambridge, MIT Press, 1992

[Gelder et al., 91] van Gelder, A., Ross, K., Schlipf, J., The Well-Founded Semantics for General Logic Programs, Journal of ACM, p.221-230, 1990

[Gelfond und Lifschitz, 88] Gelfond, M., Lifschitz, V.: Compiling Circumscriptive Theories into Logic Programs, Proc. 2nd Int. Workshop on Nonmonotonic Reasoning, Springer, LNCS 346, 1988

[Gelfond und Lifschitz, 90] Gelfond, M., Lifschitz, V., Logic Programs with Classical Negation, Proc. 7th Intl. Conference on Logic Programming, 1990

[Genesereth und Nilsson, 87] Genesereth, Michael R., Nilsson, Nils J.: Logical Foundations of Artificial Intelligence, Morgan Kaufmann, Los Altos, 1987

[Gordon, 87] Gordon, Thomas F.: Oblog 2 – A Hybrid Knowledge Representation System for Defeasible Reasoning, Proc. 1st Intl. Conference on Artificial Intelligence and Law, Boston, ACM Press, 1987

[Gordon, 95] Gordon, Thomas F., The Pleadings Game: An Artificial Intelligence Model of Procedural Justice. Kluwer, Dordrecht, 1995.

[Hanks und McDermott, 87] Hanks, Steven, McDermott, Drew: Nonmonotonic Logic and Temporal Projection, Artificial Intelligence 33, 1987

[Konolige, 88] Konolige, Kurt: On the Relation Between Default and Autoepistemic Logic, Artificial Intelligence 35 (3), 1988

[Kraus et al., 90] Kraus, Sarit, Lehmann, Daniel, Magidor, Menachem: Nonmonotonic Reasoning, Preferential Models and Cumulative Logics, Artificial Intelligence 44, 167–207, 1990

[Lakemeyer, 92] Lakemeyer, Gerhard: All You Ever Wanted to Know About Tweety, Proc. 3rd Intl. Conference on Principles of Knowledge Representation and Reasoning, Cambridge, 1992

[Levesque, 90] Levesque, Hector: All I know: A Study in Autoepistemic Logic, Artificial Intelligence 42, 263–309, 1990

[Lifschitz, 85] Lifschitz, Vladimir: Computing Circumscription, Proc. IJCAI 85, 1985

[Lifschitz, 91] Lifschitz, Vladimir: Nonmonotonic Databases and Epistemic Queries, Proc. 9th National Conference on Artificial Intelligence, 1991

[Lifschitz, 95] Lifschitz, V., Foundations of Declarative Logic Programming, in: G. Brewka (ed.), Principles of Knowledge Representation, Studies in Logic, Language and Information, CSLI publications, 1996

[Lin und Shoham, 90] Lin, Fangzhen, Shoham, Yoav: Epistemic Semantics for Fixed-Points Nonmonotonic Logics, Proc. Theoretical Aspects of Reasoning About Knowledge (TARK), 1990

[Lehmann, 92] Lehmann, Daniel: Another Perspective on Default Reasoning, Technical Report TR 92-12, Leibniz Center for Computer Science, Hebrew University, Jerusalem, 1992

[Lukaszewicz, 88] Lukaszewicz, Witold: Considerations on Default Logic, Computational Intelligence 4, 1988

[Makinson, 89] Makinson, David: General Theory of Cumulative Inference, In: Reinfrank et al. (eds), Non-Monotonic Reasoning, Springer, LNAI 346, 1989

[Makinson, 94] Makinson, David: General Patterns in Nonmonotonic Reasoning, In: Gabbay, D.M., Hogger, C.J., Robinson, J.A. (Hrsg.), Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 3, Oxford University Press, 1994

[Marek und Truszczynski, 89] Marek, Wiktor, Truszczynski, Miroslaw: Relating Autoepistemic and Default Logics, Proc. First Intl. Conference on Priciples of Knowledge Representation and Reasoning, 1989

[Marek und Truszczynski, 93] W. Marek and M. Truszczyński. *Nonmonotonic Logics – Context-Dependent Reasoning.* Springer, 1993.

[McCarthy, 80] McCarthy, John: Circumscription – A Form of Nonmonotonic Reasoning, Artificial Intelligence 13, 1980

[McCarthy, 84] McCarthy, John: Applications of Circumscription to Formalizing Common Sense Knowledge, Proc. AAAI-Workshop Non-Monotonic Reasoning, 1984 (auch in Artificial Intelligence 28, 1986)

[Moore, 85] Moore, Robert C.: Semantical Considerations on Nonmonotonic Logic, Artificial Intelligence 25, 1985 (Kurzfassung in Proc. IJCAI 83)

[Niemelä und Simons, 97] Niemelä, I., Simons, P., Efficient Implementation of the Stable Model and Well-Founded Semantics for Normal Logic Programs, Proc. 4th Intl. Conference on Logic Programming and Nonmonotonic Reasoning, Dagstuhl, Germany, Springer Verlag, 1997

[Pearl, 90] Pearl, Judea: System Z: A Natural Ordering of Defaults with Tractable Applications to Nonmonotonic Reasoning, Proc. Third Conference on Theoretical Aspects of Reasoning About Knowledge, 1990

[Pereira und Alferes, 89] Pereira, L.M., Alferes, J.J., Well Founded Semantics for Logic Programs with Explicit Negation, Proc. 10th European Conference on Artificial Intelligence, Vienna, 1992

[Poole, 88] Poole, D.: A Logical Framework for Default Reasoning, Artificial Intelligence 36, 1988

[Poole, 89] Poole, D.: What the Lottery Paradox Tells Us about Default Reasoning, Proc. 1. Intl. Conference Principles of Knowledge Representation and Reasoning, 1989

[Poole et al., 86] Poole, D., Goebel, R., Aleliunas, R.: A Logical Reasoning System for Defaults and Diagnosis, University of Waterloo, Dep. of Computer Science, Research Rep. CS-86-06, 1986

[Przymusinski, 90] Przymusinski, T., The Well-Founded Semantics Coincides with the Three-Valued Stable Semantics, Fundamenta Informaticae, 1989

[Przymusinski, 91] Przymusinski, T., Stable Semantics for Disjunctive Programs, New Generation Computing, 9:401-424, 1991

[Reiter, 80] Reiter, Raymond: A Logic for Default Reasoning, Artificial Intelligence 13, 1980

[Rescher, 64] Rescher, Nicholas: Hypothetical Reasoning, North Holland Publ., Amsterdam, 1964

[Sagonas et al., 96] Sagonas, K., Swift, T. and David S. Warren. An Abstract Machine for Computing the Well-Founded Semantics. In Proceedings of the Joint International Conference and Symposium on Logic Programming Bonn, Germany, pages 274–288,MIT Press, 1996

[Schaub, 98] Schaub, Torsten, The Automation of Reasoning with Incomplete Information: From Semantic Foundations to Efficient Computation, Springer Verlag, 1998.

[Siegel, 90] Siegel, Pierre: A Modal Language for Nonmonotonic Logic, Proc. DRUMS Workshop, Marseille, 1990

[Tarski, 55] Tarski, Alfred, A lattice-theoretical fixpoint theorem and its applications, Pacific Journal of Mathematics, 5:285-309, 1955