Logical Foundations of Artificial Intelligence

Michael R. Genesereth and Nils J. Nilsson
Stanford University

MORGAN KAUFMANN PUBLISHERS, INC.
95 FIRST STREET
LOS ALTOS, CALIFORNIA 94022

Editor and President *Michael B. Morgan*
Production Manager *Jennifer M. Ballentine*
Cover Designer *Irene Imfeld*
Composition *Arthur Ogawa and Laura Friedsam*
Book Design *Beverly Kennon-Kelley*
Copy Editor *Lyn Dupré*
Graduate Assistant *Jamison Gray*
Production Assistant *Todd R. Armstrong*

Library of Congress Cataloging-in-Publication Data

```
Genesereth, Michael, 1948-
   Logical foundations of artificial intelligence.

   Bibliography: p.
   Includes index.
   1. Artificial intelligence.    I. Nilsson, Nils,
1933-      .    II. Title.
Q335.G37  1986          006.3          87-5461
ISBN 0-934613-31-1
```

Morgan Kaufmann Publishers, Inc.
95 First Street, Los Altos, CA 94022

ISBN: 0-934613-31-1

90 89 88 87 5 4 3 2 1

CHAPTER 6
Nonmonotonic Reasoning

WE HAVE ALREADY SEEN some indication of the power of the first-order predicate calculus as a language for expressing declarative knowledge in AI systems. We can use the predicate calculus to express any conceptualization based on objects and their relations in a domain of discourse. Given what we have presented so far, we might imagine that a typical AI system using first-order logic would work somewhat as follows. Information that the system has about its domain is expressed as a finite set Δ of first-order formulas. We call Δ the database or *base set of beliefs* of the system. To answer queries or to take appropriate actions, the system typically will have to decide whether or not some formula ϕ is logically entailed by its beliefs. We can imagine that the system will make this decision by performing logical deductions on Δ, perhaps by using resolution on the clause form of $\Delta \wedge \neg \phi$. (Our notation is simplified by letting Δ also stand for the conjunction of the formulas in the set Δ.)

Even though this model is quite useful for a variety of tasks requiring knowledge about a domain, it has major limitations. The three most important ones are:

(1) Language (probably *any* language) cannot capture all that we want to say about the world. A finite set of sentences can never be more than an approximate description of things as they really are. Any general rule that we might care to frame is subject to an unlimited number of exceptions and qualifications. If we are going to use language to describe the world, we will have to use it in a way that is robust in the face of an ever-expanding set of more highly articulated statements.

(2) The inference rules of ordinary logic (such as modus ponens and the resolution principle) are *sound.* Thus, deductions from a base set of beliefs never produce *new* knowledge about the world. If ϕ logically follows from Δ, then all the models of Δ, including our intended interpretation, also are models of ϕ. Deriving ϕ in no way eliminates any of the models and thus ϕ tells us nothing about the world that was not already described by Δ. Of course, we want to be able to manipulate our description of the world so that implicit facts about it become represented explicitly, and sound inference rules do just that for us. We also will want, however, to add formulas to Δ that say new (or revised) things about the world, and ordinary logic gives us no hints about how to do that. We need methods for reasoning with tentative statements because they are the only ones we are ever going to have. These reasoning methods will have to anticipate the possibility of later revisions of the knowledge base.

(3) The logical languages that we have used so far are adequate for expressing only those statements that we are willing to take as being either wholly true or wholly false. Often, we have information about a situation that is known to be *uncertain.* For example, we know that it is likely (but not certain) that it will be sunny in Pasadena on New Year's Day.

In the next few chapters, we will be concerned with confronting and overcoming some of these limitations. An important technique involves the use of *nonsound* inferences of various kinds. That is, from a database Δ, we will allow certain inferences that do not logically follow from Δ. Often, these inferences depend globally on *all* of the sentences in Δ rather than on a small subset. In particular, we will be introducing inference techniques the application of which depends on certain sentences *not* being in Δ. With such inference rules, if another sentence is added to Δ, an inference may have to be retracted. For this reason, such inference rules are called *nonmonotonic.* Ordinary logical inference rules, on the other hand, are monotonic because the set of theorems derivable from premises is not reduced by adding to the premises.

There are many situations in which it is appropriate for intelligent systems to augment their beliefs by new ones that do not logically follow from their explicit ones. The press of events sometimes compels *some* action before all the relevant facts are at hand. It would be useful perhaps for systems to be able to assume that the beliefs they have presently about a certain subject are all the beliefs that are important about that subject. Natural-language dialogues among humans, for example, depend on both the speaker and the hearer using several general augmenting conventions of this sort. (Example: "He didn't say John was his brother, so I'll assume he is not.")

Also, as we mentioned earlier, there is a sense in which any attempt to capture *all* the knowledge about the real world by a finite set of sentences is fundamentally impossible. As our own knowledge (and that of science) increases, our conceptualization of a subject area changes. Any conceptualization, put forward for certain purposes, is subject to challenge. Consider, for example, the following sentence about birds: "All birds fly." With the obvious intended interpretation, we might express this as $\forall$x Bird(x) $\Rightarrow$ Flies(x). This sentence might be useful for certain limited purposes, but if we tried to apply it more generally, we might be confronted by the fact that ostriches, which are birds, do not fly. After having this problem pointed out to us, we could correct it by changing our axiom about birds flying to the following:

$$\forall x\ \mathtt{Bird(x)} \land \neg \mathtt{Ostrich(x)} \Rightarrow \mathtt{Flies(x)}$$

Even this sentence does not accurately capture the *real* world, however, because we can imagine several other kinds of birds that do not fly: baby birds, dead birds, wingless birds, and so on. The list of such *qualifications* is very long if not endless, leading us, perhaps, to despair of using language for knowledge representation. This problem often is called the *qualification problem.* Most universally quantified sentences will have to include an infinite number of qualifications if they are to be interpreted as accurate statements about the world. Yet, in our everyday reasoning, we humans use sentences that we assume to be true. What we would seem to need for our machines is an inference rule that permits us to make somewhat temporary, or *default*, assumptions that can later be revised when additional qualifications become important.

There are a number of ways to achieve the appropriate nonmonotonic effects. In this chapter, we explore three methods. In one, we adopt a convention that allows us to infer that, if we cannot prove a ground atom, then we can assume its negation. In another, we show how to compute a formula that can be added to Δ that restricts the objects that satisfy a certain predicate to just those that Δ says must satisfy that predicate. In a third, we introduce nonmonotonic rules of inference called *defaults*, and show how they are used to derive default conclusions.

These methods have several potential applications. Through examples in this and the next chapter, we indicate their utility for making reasonable assumptions about what can be inferred from a finite set of sentences. We see these nonmonotonic techniques as promising candidates for extending the reach of logic beyond some of the limitations we have noted.

6.1 The Closed-World Assumption

Recall that a theory $\mathcal{T}$ is *complete* if either every ground atom in the language or its negation is in the theory. Thus, the logical closure of the formula `P(A) ∧ (P(A) ⇒ Q(A)) ∧ P(B)` is not a complete theory, because neither `Q(B)` nor `¬Q(B)` is in the theory. One convention for augmenting a theory is to complete it.

The most straightforward and simple way to complete a theory is by a convention called the *closed-world assumption (CWA).* The CWA completes the theory defined by a base set of beliefs, Δ, by including the *negation* of a ground atom in the completed theory whenever that ground atom does not logically follow from Δ. The effect of the CWA is as though we augmented the base belief set with all the negative ground literals the positive versions of which cannot be deduced from Δ. The CWA is nonmonotonic because the set of augmented beliefs would shrink if we added a new positive ground literal to Δ.

We define the effects of the CWA in terms of customary logical notation. We call our belief set, Δ, the *proper axioms* of a theory. The theory, denoted by $\mathcal{T}[\Delta]$, is the closure of Δ under logical entailment. The CWA *augments* $\mathcal{T}[\Delta]$ by adding a set, Δ_{asm}, of *assumed beliefs.* The closure under logical implication of the union of these assumed beliefs with Δ comprises the CWA-augmented set of beliefs CWA[Δ]. The CWA can be stated succinctly as follows:

- The formula ϕ (constructed from elements of a predefined predicate-calculus language) is in $\mathcal{T}[\Delta]$ if and only if $\Delta \models \phi$. (This is the ordinary definition of a theory $\mathcal{T}[\Delta]$ in terms of a base set Δ.)
- $\neg P$ is in Δ_{asm} if and only if the ground atom P is not in $\mathcal{T}[\Delta]$. (Δ_{asm} is a set of added beliefs assumed under the CWA.)
- ϕ is in CWA[Δ] if and only if $\{\Delta \cup \Delta_{asm}\} \models \phi$. (The augmented theory, CWA[Δ], is the closure of all beliefs, explicit and assumed.)

In our example, in which Δ is `P(A) ∧ (P(A) ⇒ Q(A)) ∧ P(B)`, the CWA adds `¬Q(B)`, since Δ does not logically entail `Q(B)`.

The CWA often is used with database systems. Suppose we have a database listing pairs of countries that are geographic neighbors:

```
Neighbor(US,Canada)

Neighbor(US,Mexico)

Neighbor(Mexico,Guatemala)

          ⋮
```

For such a database, it would be useful to adopt the convention that countries that are not listed as neighbors are not neighbors. That convention is an example of the CWA. Without such a convention, we would have to list explictly all the nonneighbor pairs if we wanted to be able to answer queries such as, “Are Brazil and Canada neighbors?”

Note that the CWA depends on a syntactic feature of a set of beliefs; namely, whether a *positive* ground literal can be derived. If we systematically replaced each predicate letter P_i by $\neg Q_i$ (defining $P_i \equiv \neg Q_i$) the theory would be the same, but the CWA would give different results with respect to the original predicates. The convention is most efficient when the "positive facts" are few in number compared to the "negative facts." A database designer who uses the CWA will want to conceptualize the domain in a way that matches this expectation.

We might ask whether the CWA always results in a consistent augmented theory, CWA[Δ]. The following example shows that it does not.

Let Δ contain only the clause `P(A) ∨ P(B)`. Then neither `P(A)` nor `P(B)` is in $\mathcal{T}[\Delta]$, so by the CWA their negations are *both* in CWA[Δ]. Together, however, these negations are not consistent with `P(A) ∨ P(B)`.

The source of this difficulty is that Δ contained a disjunction of ground atoms (positive ground literals) but no way to prove any of them. Thus, the conjunction of their negations, which contradicts the original disjunction, is in the augmented theory. The following theorem links this difficulty with the inconsistency of CWA[Δ].

THEOREM 6.1 CWA[Δ] *is consistent if and only if, for every positive-ground-literal clause $L_1 \vee L_2 \vee \ldots \vee L_n$ that follows from Δ, there is also entailed by Δ at least one ground literal L_i that subsumes it. (Equivalently, the CWA augmentation* CWA[Δ] *of a consistent Δ is inconsistent if and only if there are positive ground literals $L_1, \ldots, L_n$ such that $\Delta \models L_1 \vee L_2 \vee \ldots \vee L_n$ but, for $i = 1, \ldots, n$, $\Delta \not\models L_i$.)*

Proof CWA[Δ] can be inconsistent only if $\Delta \cup \Delta_{asm}$ is. Then, by the compactness theorem of logic, there is a finite subset of Δ_{asm} that contradicts Δ. Let this subset be $\{\neg L_1, \ldots, \neg L_n\}$. Then Δ implies the negation of the conjunction of these formulas; that is, $\Delta \models L_1 \vee \ldots \vee L_n$. Since each $\neg L_i$ is in Δ_{asm}, by the definition of Δ_{asm}, none of the L_i follow from Δ. The proof in the other direction is obvious. □

The application of Theorem 6.1 depends critically on the terms that we allow as part of the language. For example, if the only object constants in the language are `A` and `B`, then the following clauses do not have an inconsistent augmentation (even though one of them is a disjunction of positive literals):

```
P(x) ∨ Q(x)

P(A)

Q(B)
```

In this case, the only ground clauses of the form $L_1 \vee L_2 \vee \ldots \vee L_n$ that can be proved from Δ are P(A) ∨ Q(A) and P(B) ∨ Q(B) (by universal instantiation). Each of these is subsumed by one of the clauses in Δ. On the other hand, if we also admit the object constant C, we also can prove P(C) ∨ Q(C), but we can prove neither P(C) nor Q(C) to subsume it, so the CWA produces an inconsistent augmentation in that case.

In the first case of this example, we limited the object constants of the language to those that occurred in Δ. We sometimes also want to make the assumption that the only objects in the domain are the ones that can be named using the object and function constants occurring in the language. This is called the *domain-closure assumption (DCA)*. If there are no function constants in the language, the DCA can be written as the following axiom, the domain-closure axiom:

$$\forall x\ x{=}t_1 \vee x{=}t_2 \vee \ldots$$

where the t_i are the object constants of the language. (If the language contained function constants, there would be an infinite number of terms that could be constructed, and typically the DCA could not be expressed by a first-order formula.) This axiom makes a strong assumption. It allows us, for example, to replace any quantifiers by finite conjunctions and disjunctions; then the belief set would be equivalent to a propositional combination of ground literals.

Another assumption often used in connection with nonmonotonic reasoning is the *unique-names assumption (UNA)*: If ground terms cannot be proved equal, they can be assumed unequal. The UNA is a consequence of the CWA; it is merely an application of the CWA to the equality predicate. The DCA is sometimes used in addition to the CWA to restrict the augmentation further.

Since it may be difficult to test the conditions of Theorem 6.1, the following corollary is important. (Recall that a *Horn clause* is defined to be one that has at most one positive literal.)

COROLLARY 6.1 *If the clause form of* Δ *is Horn and consistent, then the CWA augmentation* CWA[Δ] *is consistent.*

Proof Suppose the contrary; that is, that Δ is Horn and consistent but that CWA[Δ] is inconsistent. Then, according to Theorem 6.1, we can deduce from Δ a ground clause $L_1 \vee L_2 \vee \ldots \vee L_n$ containing only positive ground literals no one of which is derivable from Δ. Thus, $\Delta \cup \{\neg L_1, \ldots, \neg L_n\}$ is inconsistent. However, because Δ contains only Horn clauses, it must then be the case that for some i, $\Delta \wedge \neg L_i$ is inconsistent (see Exercise 3). Or, for some i, $\Delta \models L_i$. But this contradicts the choice of L_i. □

Thus we see that an important class of theories—the so-called Horn theories—have consistent CWA augmentations. We see from Theorem 6.1, however, that the condition that Δ be Horn is not absolutely necessary for the CWA augmentation of Δ to be consistent.

The CWA is too strong for many applications. We do not always want to assume that *any* ground atom not provable from Δ is false. Weakening this assumption in the natural way leads to the idea of the CWA *with respect to a predicate P*. Under that convention, ground atoms in some particular predicate, *P*, that are not provable from Δ are assumed to be false. The assumed beliefs, Δ_{asm}, in that case contain only negative ground literals in *P*.

For example, suppose Δ is:

$$\forall x\ Q(x) \Rightarrow P(x)$$

$$Q(A)$$

$$R(B) \vee P(B)$$

Applying the CWA to Δ with respect to P allows us to conclude $\neg P(B)$, because P(B) cannot be concluded from Δ. That, in turn, allows us also to conclude R(B) from Δ. (Unconstrained application of the CWA to Δ would have condoned concluding both $\neg R(B)$ and $\neg P(B)$, which contradict Δ.)

We also can make the CWA with respect to a *set* of predicates. In database applications, this assumption allows us to assume that certain relations in the database are complete and others are not. If the set contains all the predicates in Δ, then we get the same result as we would have got with the ordinary CWA.

It is interesting to note that the CWA with respect to a set of predicates may produce an inconsistent augmentation even when the CWA with respect to each one of the predicates alone produces a consistent augmentation. For example, the CWA with respect to the set $\{P,Q\}$ is inconsistent with the belief set $(P \vee Q)$, even though the CWA with respect to either P or Q is consistent with that belief set.

We might be tempted to say that the source of this difficulty is that $(P \vee Q)$ is not Horn in the set $\{P,Q\}$. (We say that a set of clauses is Horn in the predicate *P* if there is at most one positive occurrence of *P* in each clause. We say that a set Δ of clauses is Horn in a set of predicates Π if and only if each of the clauses would be Horn in *P* after substituting the letter *P* for each occurence of any of the letters in Π in the clauses of Δ.) Even if a belief set were Horn in a set of predicates, however, the CWA with respect to the predicates in this set might produce an inconsistent augmentation. Suppose Δ is $\{P(A) \vee Q,\ P(B) \vee \neg Q\}$. This set is Horn in the set $\{P\}$, and making the CWA with respect to the predicates (just P) in $\{P\}$ yields both $\neg P(A)$ and $\neg P(B)$. These, together with Δ, are inconsistent.

6.2 Predicate Completion

It happens that we often can express in a single sentence of logic the assumption that the only objects that satisfy a predicate are those that *must* do so—given our beliefs. We will describe several of these methods—all related but of increasing generality and power.

First consider the simple case in which P(A) is the *only* formula in Δ. P(A) is equivalent to the following expression:

$$\forall x\ x=A \Rightarrow P(x)$$

Such a formula could be taken to be the "if" half of a *definition* for P. The assumption that there are no other objects that satisfy P can then be made by writing the "only if" half as

$$\forall x\ P(x) \Rightarrow x=A$$

This half is called the *completion formula* for P. It makes the explicit information about P in Δ *complete.*

The conjunction of Δ with the completion formula is called *the completion of* P *in* Δ, and is denoted by COMP[Δ; P]. In this case, we have

$$\begin{aligned} \mathrm{COMP}[\Delta;P] &\equiv (\forall x\ P(x) \Rightarrow x=A) \wedge \Delta \\ &\equiv \forall x\ P(x) \Leftrightarrow x=A \end{aligned}$$

In this example, predicate completion (if augmented by the UNA) produces the same effect as the CWA with respect to P.

If Δ contained only two formulas in P, say P(A) and P(B), the completion formula would be

$$\forall x\ P(x) \Rightarrow x=A \vee x=B$$

Here again, predicate completion of P (together with unique names) has the same effect as the CWA with respect to P.

If Δ contains formulas in which a predicate P occurs disjunctively with other predicates or in which P contains variables, predicate completion is more complex. In fact, we define predicate completion for only certain kinds of clauses.

We say that a set of clauses is *solitary* in P if each clause with a positive occurrence of P has at most *one* occurrence of P. Note that clauses solitary in P are also Horn in P, but not necessarily vice versa. For example $Q(A) \vee \neg P(B) \vee P(A)$ is Horn in P, but not solitary in P.

$\{\varphi_1, \varphi_2, \psi\}$

$\neg(\varphi_1 \vee \varphi_2) \supset \psi$

$\neg\varphi_1 \wedge \neg\varphi_2 \supset$

φ

17
0

We define predicate completion of P only for clauses solitary in P. Suppose Δ is a set of clauses solitary in P. We can write each of the clauses in Δ that contains a positive P literal in the following form:

because

$\neg\varphi \vee \psi$ ∴

$\varphi \supset \psi$

$$\forall \mathrm{y}\ Q_1 \wedge \ldots \wedge Q_m \Rightarrow P(t)$$

where t is a tuple of terms, $[t_1, t_2, \ldots, t_n]$, and the Q_i are literals not containing P. There may be no Q_i, in which case the clause is simply $P(t)$. The Q_i and t may contain variables, say the tuple of variables y.

This expression is equivalent to

$$\forall \mathrm{y} \forall \mathrm{x}\ (\mathrm{x}{=}t) \wedge Q_1 \wedge \ldots \wedge Q_m \Rightarrow P(\mathrm{x})$$

where x is a tuple of variables not occurring in t and $(\mathrm{x}{=}t)$ is an abbreviation for $(\mathrm{x}_1{=}t_1 \wedge \ldots \wedge \mathrm{x}_n{=}t_n)$. Finally, since the variables y occur in only the antecedent of the implication, this expression is equivalent to

$$\forall \mathrm{x}\ (\exists \mathrm{y}\ (\mathrm{x}{=}t) \wedge Q_1 \wedge \ldots \wedge Q_m) \Rightarrow P(\mathrm{x})$$

This way of writing the clause is called the *normal form* of the clause. Suppose there are exactly k clauses ($k > 0$) in Δ that have a positive P literal. Let the normal forms of these clauses be

$$\begin{gathered}\forall \mathrm{x}\ E_1 \Rightarrow P(\mathrm{x}) \\ \forall \mathrm{x}\ E_2 \Rightarrow P(\mathrm{x}) \\ \vdots \\ \forall \mathrm{x}\ E_k \Rightarrow P(\mathrm{x})\end{gathered}$$

Each of the E_i will be an existentially quantified conjunction of literals, as in the preceding generic case. If we group together these clauses as a single implication, we obtain

$$\forall \mathrm{x}\ E_1 \vee E_2 \vee \ldots \vee E_k \Rightarrow P(\mathrm{x})$$

Here we have an expression that can be taken as the "if" half of a definition for P. It suggests the following, "only if," *completion formula* for P:

$$\forall \mathrm{x}\ P(\mathrm{x}) \Rightarrow E_1 \vee E_2 \vee \ldots \vee E_k$$

Since the E_i do not contain P, the "if" and "only if" parts together can be thought of as a *definition* for P:

$\forall x$ $\forall x$

$$\forall x\ P(x) \Leftrightarrow E_1 \vee E_2 \vee \ldots \vee E_k$$

$\neg Ox \vee Bx$

$\{\neg Ox, Bx\}$

$\{B(t)\}$

$\{\neg O(s)\}$

Since the "if" part already is entailed by Δ, we define the *completion of P in* Δ as $B\,t$ $\forall x\ x=t \supset Bx$

$$\text{COMP}[\Delta; P] \equiv_{def} \Delta \wedge (\forall x\ P(x) \Leftrightarrow E_1 \vee \ldots \vee E_k)$$

where the E_i are the antecedents of the normal forms of the clauses in Δ, as defined previously.

Let us consider a simple example of predicate completion. Suppose Δ is

$Ox \rightarrow Bx$ $\neg Ox \vee Bx$

```
∀x Ostrich(x) ⇒ Bird(x)     {O
Bird(Tweety)                {Bird(Tweety)}
¬Ostrich(Sam)               {¬Ostrich(Sam)}
```

(All ostriches are birds; Tweety is a bird; Sam is not an ostrich.) We note that Δ is solitary in Bird. Let us complete Bird in Δ. Writing those clauses containing Bird in normal form yields

$\forall x\ \neg Ox \vee Bx$

$\{\neg Ox, Bx\}$

$$\forall x\ \text{Ostrich}(x) \vee x{=}\text{Tweety} \Rightarrow \text{Bird}(x)$$

$\forall x\ x=t \supset B$

$\forall x\ \neg Ox \wedge x=$

$\forall x\ x=t \vee \neg Ox \supset B$

The completion of Bird in Δ is then simply

$\forall x$

$$\text{COMP}[\Delta;\text{Bird}] \equiv \Delta \wedge (\forall x\ \text{Bird}(x) \Leftrightarrow \text{Ostrich}(x) \vee x{=}\text{Tweety})$$

$\forall x\ x=\text{tweety}$

$\forall x$

(The only birds are ostriches or Tweety.) With the completion formula added to Δ (and augmented by the UNA) we could prove, for example, ¬Bird(Sam).

$\forall x$

$\forall x$

What is predicate completion doing for us in this case? Δ tells us that Tweety is a bird, that Sam is not an ostrich, and that all ostriches are birds. Completion of Bird in Δ is a way of making the assumption that there are no birds other than those about which Δ tells us. That is, the only birds are Tweety and ostriches. Since Sam is not an ostrich, and since the UNA lets us assume that Sam is not Tweety, we can conclude that Sam is not a bird.

If we did not limit Δ to clauses solitary in P, the completion process might produce circular definitions for P, which would not restrict the objects that satisfy P to those that must do so, given Δ. Sometimes, we can formally apply the completion process to clauses Horn (but not solitary) in P and still get meaningful results. Consider the following Horn clauses that describe the factorial relation (we assume implicit universal quantification):

$\forall x\,(\neg Dx \wedge x = \neg w) \supset Bx$

$(Dx \quad \vee \neg -$

```
x=0 ⇒ Factorial(x,1)

x≠0 ∧ Factorial(Minus(x,1),y) ⇒ Factorial(x,Times(x,y))
```

Writing these expressions in normal form yields

```
x=0 ∧ z=1 ⇒ Factorial(x,z)

(∃y x≠0 ∧ z=Times(x,y) ∧ Factorial(Minus(x,1),y)) ⇒
  Factorial(x,z)
```

Now we formally perform predicate completion on `Factorial` (even though the clauses are not solitary in `Factorial`). The result is

$\forall x \quad \exists$

```
Factorial(x,z) ⇒
  (x=0 ∧ z=1) ∨
  (∃y x≠0 ∧ z=Times(x,y) ∧ Factorial(x-1,y))
```

This result is easily interpreted as a recursive definition of factorial. It illustrates that limiting predicate completion to solitary clauses is perhaps unnecessarily restrictive. Not all definitions of a predicate in terms of itself are circular—some are recursive.

There are two special cases of predicate completion that give rise to interesting forms for the completion formulas. Suppose Δ is of the form (∀x P(x)). Making use of the atom T, we can write this clause as (∀x T⇒P(x)). Completion of P then gives the completion formula (∀x P(x)⇒T), which is a valid formula and thus does not further restrict our theory. (Restricting the objects that satisfy P to all objects in the domain is no restriction.)

On the other hand, if there are *no* clauses in Δ that are positive in P, we can assume any valid one; e.g., (∀x F⇒P(x)). Completion of P then gives the completion formula (∀x P(x)⇒F), which is equivalent to (∀x ¬P(x)). In this latter case, Δ says nothing about there being any objects satisfying P; therefore, we assume there are none.

Although in simple cases predicate completion and the CWA have the same effect, in general they are different. For example, suppose that Δ contains the single formula P(A), and that the language also contains the object constant B. Then the CWA extension includes ¬P(B), and the completion formula is (∀x P(x)⇒(x=A)). These two expressions are not equivalent, although ¬P(B) plus the DCA entails (∀x P(x)⇒(x=A)); and (∀x P(x)⇒(x=A)) plus the UNA entails ¬P(B). ([Lifschitz 1985b] derives general conditions relating these two augmenting conventions.)

Predicate completion, like the CWA, is nonmonotonic because, if another clause positive in P were added to Δ, the completion formula for P would be different. In general, it would be weaker; i.e., the augmented theory would allow more objects to satisfy P than the original one did. Thus, some proofs of expressions of the form $\neg P$ could no longer be obtained. In our earlier example about birds, if we were to augment Δ by including `Penguin(x)`$\Rightarrow$`Bird(x)`, then the new completion formula for `Bird` would be:

$$\texttt{Bird(x)} \Rightarrow \texttt{Ostrich(x)} \vee \texttt{Penguin(x)} \vee \texttt{x=Tweety}$$

and we could no longer prove $\neg$`Bird(Sam)` as we could earlier. (Sam might be a penguin.)

Extending a set of beliefs by predicate completion preserves their consistency.

THEOREM 6.2 *If Δ is a consistent set of clauses solitary in P, then the completion of P in Δ is consistent.*

This theorem follows from stronger results, Theorem 6.7 or Theorem 6.8, given later in the chapter (also without proof).

We also can perform predicate completion of several predicates in parallel. In *parallel predicate completion* of a set of predicates, each predicate in the set is completed separately (without regard for the others), and the conjunction of these separate completion formulas is added to Δ. The completion process for each uses only the original clauses in Δ and not the formulas added by the completion process for other predicates. Parallel predicate completion allows us to restrict the objects that satisfy any of several predicates to those that are forced to do so by Δ.

For the several completion formulas to avoid circularity, we must impose a condition on the way in which the predicates being completed can occur in Δ. To motivate this condition, consider the following clauses (which are solitary in `P`, `Q`, and `R`):

$$\texttt{Q(x)} \Rightarrow \texttt{P(x)} \qquad \neg Q(x) \vee P(x)$$

$$\texttt{R(x)} \Rightarrow \texttt{Q(x)} \qquad \neg R(x) \vee Q(x)$$

$$\texttt{P(x)} \Rightarrow \texttt{R(x)} \qquad \neg P(x) \vee R(x)$$

Parallel predicate completion of {`P`,`Q`,`R`} would yield:

$$\texttt{P(x)} \Leftrightarrow \texttt{Q(x)} \Leftrightarrow \texttt{R(x)} \Leftrightarrow \texttt{P(x)}$$

which is circular.

Recall that writing clauses that are solitary in P in their normal forms allowed us to combine all clauses in Δ containing a positive P literal into a single formula of the form

$$\forall \mathtt{x}\ E_1 \vee E_2 \vee \ldots \vee E_k \Rightarrow P(\mathtt{x})$$

Denoting the antecedent of this implication simply by E gives us

$$\forall \mathtt{x}\ E \Rightarrow P(\mathtt{x})$$

where E contains *no* occurrences of P.

To perform parallel predicate completion of the set $\Pi = \{P_1, P_2, \ldots, P_n\}$ of predicates in Δ, we first write those clauses in Δ containing members of Π in their normal forms, then combine the clauses containing the same P_is into single formulas.

$$\begin{gathered}\forall \mathtt{x}\ E_1 \Rightarrow P_1(\mathtt{x})\\ \forall \mathtt{x}\ E_2 \Rightarrow P_2(\mathtt{x})\\ \forall \mathtt{x}\ E_3 \Rightarrow P_3(\mathtt{x})\\ \vdots\\ \forall \mathtt{x}\ E_n \Rightarrow P_n(\mathtt{x})\end{gathered}$$

Parallel predicate completion is then accomplished by adding to Δ the completion formulas $(\forall \mathtt{x}\ P_i(\mathtt{x}) \Rightarrow E_i)$ for $i = 1, \ldots, n$. To avoid circular definitions of the P_i, we must be able to order the P_i such that each E_i has no occurrences of any of $\{P_i, P_{i+1}, \ldots, P_n\}$ (and has no negative occurrences of any of $\{P_1, \ldots, P_{i-1}\}$). If this ordering can be achieved, we say that the clauses in Δ are *ordered in* Π. In the next section, we illustrate parallel predicate completion with an example.

Note that, if Δ is ordered in Π, it also is solitary in each of the P_i individually (but not necessarily conversely).

Theorem 6.2, about the consistency of predicate completion, can be generalized to the case of parallel predicate completion.

THEOREM 6.3 *If Δ is consistent and ordered in Π, then the parallel completion of Π in Δ is consistent.*

This theorem is a consequence of extended versions of either Theorem 6.7 or Theorem 6.8, given later in the chapter.

6.3 Taxonomic Hierarchies and Default Reasoning

Several AI systems have included simple mechanisms to allow a kind of reasoning called *reasoning by default.* Since typically birds can fly, for example, we assume (by default) that an arbitrary bird can fly unless we know that it cannot. In this section, we describe a technique for stating the *typical* properties of objects and then show how a variant of parallel predicate completion can be used to perform default reasoning.

Often, this style of reasoning is applied to taxonomic hierarchies in which subclasses *inherit* the properties of their superclasses unless these properties are specifically canceled. Suppose, for example, that our beliefs, Δ, include the following formulas that define a taxonomic hierarchy:

$$\texttt{Thing(Tweety)}$$

$$\texttt{Bird(x)} \Rightarrow \texttt{Thing(x)}$$

$$\texttt{Ostrich(x)} \Rightarrow \texttt{Bird(x)}$$

$$\texttt{Flying-Ostrich(x)} \Rightarrow \texttt{Ostrich(x)}$$

(Tweety is a thing; all birds are things; all ostriches are birds; all flying ostriches are ostriches.)

This subset of Δ that defines the taxonomic hierarchy will be denoted by Δ_H.

Suppose we also want to include in Δ statements that describe some of the properties of the objects in the taxonomic hierarchy. For example, we might want to say that no things except birds can fly, that all birds except ostriches can fly, and that no ostriches except flying ostriches can fly. One way to do this is with the following formulas:

a. $\texttt{Thing(x)} \wedge \neg\texttt{Bird(x)} \Rightarrow \neg\texttt{Flies(x)}$
b. $\texttt{Bird(x)} \wedge \neg\texttt{Ostrich(x)} \Rightarrow \texttt{Flies(x)}$
c. $\texttt{Ostrich(x)} \wedge \neg\texttt{Flying-Ostrich(x)} \Rightarrow \neg\texttt{Flies(x)}$
d. $\texttt{Flying-Ostrich(x)} \Rightarrow \texttt{Flies(x)}$

The subset of Δ that describes properties of objects in the hierarchy will be denoted by Δ_P. Whether we regard a predicate as defining a taxonomic *kind* of object or a nontaxonomic *property* of an object is left to us. In this example, we choose to think of flying simply as a property that certain objects have—not as defining a kind of object.

Here, exceptions to general rules are listed explicitly in the rules. If we had exceptions to birds flying other than ostriches, we would have to list each of these exceptions in rule b. Of course, a general commonsense reasoning system would need to know about other common exceptions, such as penguins and baby birds. As we mentioned earlier in discussing the qualification problem, there would be no particular difficulty in principle with listing all known exceptions in the rule. The problem is that the system designer cannot really think of *all* of the exceptions that the system might later confront—exceptions such as wingless eagles, brain-damaged gulls, and roast ducks. Instead of listing all these exceptions, we would prefer some technique that allows us to say that birds (typically) can fly unless they are abnormal in some respect—an abnormality shared by ostriches, penguins, and such. Exceptions that we think of later can then be simply introduced by conferring this same abnormality on the new exceptions. Similarly, we would want to say that things (typically) cannot fly unless they are abnormal in some other respect—an abnormality shared by birds, airplanes, and mosquitoes. A hierarchy of exceptions would thus have to deal with several different kinds of abnormalities. We make these abnormalities part of the taxonomic hierarchy.

The following rule seems to capture what we want to say about things in general:

$$\texttt{Thing(x)} \wedge \neg\texttt{Ab1(x)} \Rightarrow \neg\texttt{Flies(x)}$$

where Ab1 is a predicate that has to do with that particular type of abnormality that must be provably absent if we are to use this general rule to prove that things cannot fly. Thus, our rule states that things do not fly unless they have an abnormality of type 1, say. (We will soon introduce other types of abnormalities.)

Birds are among those objects that have an abnormality of type 1:

$$\texttt{Bird(x)} \Rightarrow \texttt{Ab1(x)}$$

We call this rule an *inheritance cancellation rule.* The taxonomic rule Bird(x) ⇒ Thing(x) ordinarily could be used to conclude that birds *inherit* the traits of things generally—including the inability to fly (if they are not abnormal). Cancellation rules, declaring abnormalities, thus block the inheritance of specified traits. We include them in Δ_H, the formulas that define the taxonomic hierarchy.

The designer of a commonsense reasoning system could put in such information as is available about objects that might have abnormalities of type 1; e.g., airplanes, certain insects, and so on. The important feature of this way of dealing with exceptions is that additional axioms about abnormalities can be added at any time. New knowledge about flying objects can be expressed by *adding* axioms to the belief set instead of by having to *change* them!

Continuing with our example, we express the general knowledge that birds (typically) can fly by the rule:

$$\texttt{Bird(x)} \wedge \neg\texttt{Ab2(x)} \Rightarrow \texttt{Flies(x)}$$

The predicate Ab2 is about a type of abnormality the presence of which in birds prevents us from using this rule to conclude that those birds can fly. Ostriches are among those objects that have this type of abnormality; thus, we have another cancellation rule:

$$\texttt{Ostrich(x)} \Rightarrow \texttt{Ab2(x)}$$

Ordinarily, ostriches cannot fly:

$$\texttt{Ostrich(x)} \wedge \neg\texttt{Ab3(x)} \Rightarrow \neg\texttt{Flies(x)}$$

The predicate Ab3 is about a type of abnormality the presence of which in ostriches prevents us from using this rule to conclude that those ostriches cannot fly. Flying ostriches (if there are such) are among those objects having this type of abnormality:

$$\texttt{Flying-Ostrich(x)} \Rightarrow \texttt{Ab3(x)}$$

Using this approach, we now have the following formulas in Δ_H defining the taxonomic hierarchy:

$$\texttt{Flying-Ostrich(x)} \Rightarrow \texttt{Ostrich(x)}$$

$$\texttt{Flying-Ostrich(x)} \Rightarrow \texttt{Ab3(x)}$$

$$\texttt{Ostrich(x)} \Rightarrow \texttt{Bird(x)}$$

$$\texttt{Ostrich(x)} \Rightarrow \texttt{Ab2(x)}$$

$$\texttt{Bird(x)} \Rightarrow \texttt{Thing(x)}$$

$$\texttt{Bird(x)} \Rightarrow \texttt{Ab1(x)}$$

$$\texttt{Thing(Tweety)}$$

(We include the information that Tweety is a "thing" to illustrate how our approach can be used to reason nonmonotonically about the properties of Tweety.)

This taxonomy is represented graphically by the network of Figure 6.1. Note that our taxonomy does not have to be a tree. (To allow us to use parallel predicate completion, which we will do, our taxonomy does have to be a partial order.)

The properties of objects in the hierarchy are described by the following formulas in Δ_P:

$$\texttt{Thing(x)} \wedge \neg\texttt{Ab1(x)} \Rightarrow \neg\texttt{Flies(x)}$$

$$\texttt{Bird(x)} \wedge \neg\texttt{Ab2(x)} \Rightarrow \texttt{Flies(x)}$$

```
Ostrich(x) ∧ ¬Ab3(x) ⇒ ¬Flies(x)
Flying-Ostrich(x) ⇒ Flies(x)
```

We now perform parallel predicate completion on the set {`Ab1, Ab2, Ab3, Flying-Ostrich, Ostrich, Bird, Thing`} in just Δ_H to make the assumption that the only objects that are things, birds, ostriches, flying ostriches, or objects that are abnormal in any respect are those that are forced to be so by Δ_H. The clauses in Δ_H have an ordering in the set {`Ab1, Ab2, Ab3, Flying-Ostrich, Ostrich, Bird, Thing`}, so parallel predicate completion will not result in circular definitions.

In this simple example, we obtain the following completion clauses (by completing {`Ab1, Ab2, Ab3, Flying-Ostrich, Ostrich, Bird, Thing`}, in Δ_H):

1. `Thing(x) ⇒ Bird(x) ∨ x=Tweety`
2. `Bird(x) ⇒ Ostrich(x)`
3. `Ostrich(x) ⇒ Flying-Ostrich(x)`
4. `¬Flying-Ostrich(x)`
5. `Ab1(x) ⇒ Bird(x)`
6. `Ab2(x) ⇒ Ostrich(x)`
7. `Ab3(x) ⇒ Flying-Ostrich(x)`

The only object mentioned is Tweety, and it is a thing, so these clauses appropriately tell us that there are no things other than Tweety, and no birds, ostriches, or flying ostriches at all. Also, there are no objects

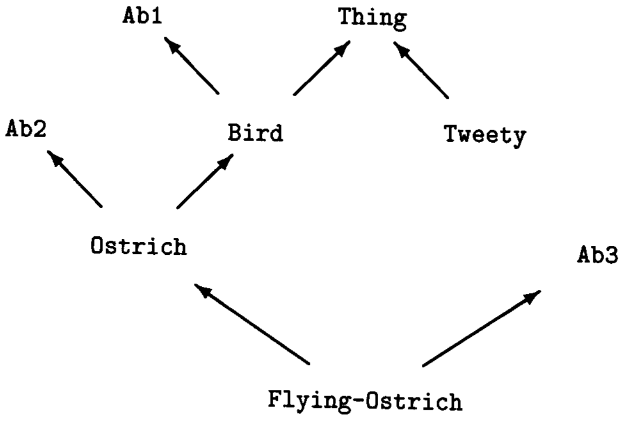

Figure 6.1 A taxonomic hierarchy with abnormalities.

abnormal in any respect. Using the properties mentioned in Δ_P, we can deduce ¬Flies(Tweety) after first proving ¬Flying-Ostrich(Tweety), ¬Ostrich(Tweety), ¬Bird(Tweety), and ¬Ab1(Tweety).

If we were to add Bird(Tweety) to our taxonomic hierarchy, completion formula 2 would change to Bird(x)⇒Ostrich(x)∨(x=Tweety). We would still be able to prove ¬Ab2(Tweety) (but not ¬Ab1(Tweety)), so we would then be able to conclude Flies(Tweety), and so on. As the reasoning system learns about other objects and other ways in which objects might have various types of abnormalities, the taxonomy changes, the predicate completion formulas are recomputed as appropriate, and the conclusions that the system can derive change correspondingly.

We call this process of completing predicates in a subset of Δ *delimited completion.* It is important to note that delimited completion of a set of predicates is not generally the same as completing these same predicates in the full Δ. (The reader should work out full completion in this case as an exercise.) Delimited completion typically produces a stronger augmenting assumption than would be produced by completion of the same predicates in the full Δ, but it is an assumption that is often justified and useful. One must be careful, however, because delimited completion might produce an inconsistent augmentation (see Exercise 6 at the end of the chapter). Later, we shall discuss a more general and robust procedure for augmenting beliefs with default assumptions of this kind.

6.4 Circumscription

Reviewing what we have said about augmentation conventions so far, we have seen that the CWA augments a belief set to include the negations of ground positive literals when these ground atoms cannot be proved true. Predicate completion is defined for belief sets consisting of clauses solitary in a predicate and augments such belief sets by formulas that state that the only objects satisfying the predicate are those that must do so, given the belief set.

Both of these augmenting ideas are based on a kind of *minimization* principle. In the case of predicate completion, the minimality idea is particularly clear. If that part of Δ containing a predicate P to be completed is written as $(\forall x)[E \Rightarrow P(x)]$, then P is *completed* by the formula $(\forall x)[P(x) \Rightarrow E]$. That is, no objects have property P unless Δ says they must.

We might like to use the same minimality assumption (i.e., that the only objects satisfying P are those that must, given Δ) in cases where Δ cannot be written as a set of clauses solitary in a predicate. For example, suppose Δ consists of the single formula $(\exists y\ P(y))$. What can be said about the smallest set of objects satisfying P in this case? This formula is not a clause, so we cannot use predicate completion. We know that there must be at least *one* object satisfying P, but there is nothing in Δ that says that there need be more than one. We can assume there is only one by adding the formula $(\exists y \forall x\ (x=y) \Leftrightarrow P(x))$.

Now, suppose Δ consists of the single clause $(P(A) \lor P(B))$. This clause is not solitary in P, so we cannot use predicate completion here either. Intuitively, however, it would seem that the formula

$$(\forall x\ P(x) \Leftrightarrow x=A) \lor (\forall x\ P(x) \Leftrightarrow x=B)$$

says what we want to say about P being minimized.

To derive augmentations of this sort for arbitrary belief sets, we must delve into the minimization process in more detail. In doing so, we define a process called *circumscription* that, like predicate completion, involves computing a certain formula that, when conjoined to Δ, says that the only objects satisfying a predicate are those that have to do so, given Δ.

Circumscription is based on the idea of *minimal* models. Let $M[\Delta]$ and $M^*[\Delta]$ be two models of Δ. (You may want to refer to Chapter 2 to review the definition of a model in logic.) We say that $M^*[\Delta]$ is no larger than $M[\Delta]$ in predicate P, writing $M^*[\Delta] \preceq_P M[\Delta]$, if (1) M and M^* have the same domain of objects, (2) all other relation and function constants in Δ besides P have the same interpretations in M and M^*, but (3) the extension of (the relation corresponding to) P in M^* is a subset of the extension of P in M. That is, if $M^* \preceq_P M$, then the set of objects satisfying P in M^* is a subset of those satisfying P in M. We write $M^* \prec_P M$ if $M^* \preceq_P M$ and $M \not\preceq_P M^*$.

There may be models of Δ that are *minimal in P* according to the ordering $\preceq_P$. M_m is *P-minimal* if, for any $M \preceq_P M_m$, $M = M_m$. (As we shall see later, minimal models do not always exist.) If a model M_m of Δ is P-minimal, then no objects satisfy the extension of P except those that have to do so, given Δ. We would like to find a sentence ϕ_P such that, for any M that is a model of $\Delta \land \phi_P$, there is no M^* that is a model of Δ with $M^* \prec_P M$. That is, the models of $\Delta \land \phi_P$ are P-minimal models of Δ. The sentence ϕ_P when conjoined to Δ asserts that there are no objects that satisfy P except those that have to do so, given Δ. We call this conjunction the *circumscription of P in* Δ.

To find an expression for ϕ_P in terms of P and Δ, we reason as follows. Let P^* be a relation constant of the same arity as P, and let $\Delta(P^*)$ be Δ with each occurrence of the relation constant P in Δ replaced by P^*. We note that any model of

$$(\forall x\ P^*(x) \Rightarrow P(x)) \land \neg(\forall x\ P(x) \Rightarrow P^*(x)) \land \Delta(P^*)$$

is *not* a P-minimal model of Δ because, in such a model, the extension corresponding to P^* would be a strict subset of the extension corresponding to P (and P^* satisfies Δ). (Again, for brevity, we allow x to be possibly a tuple of variables.) Therefore, any model of

$$\neg((\forall x\ P^*(x) \Rightarrow P(x)) \wedge \neg(\forall x\ P(x) \Rightarrow P^*(x)) \wedge \Delta(P^*))$$

is a P-minimal model of Δ.

Since P^* in the preceding expression can be *any* relation constant of the same arity as P, the ϕ_P that we seek is the second-order logic formula obtained by universally quantifying over the relation variable P*:

$$\forall P{*}\ \neg((\forall x\ P{*}(x) \Rightarrow P(x)) \wedge \neg(\forall x\ P(x) \Rightarrow P{*}(x)) \wedge \Delta(P{*}))$$

We call this expression the *circumscription formula* for P in Δ. Any model of the circumscription formula is a P-minimal model of Δ. Conjoining the circumscription formula with Δ itself gives us the *circumscription* of P in Δ:

$$\mathrm{CIRC}[\Delta; P] \equiv_{def} \Delta \wedge \forall P{*}\ \neg((\forall x\ P{*}(x) \Rightarrow P(x)) \wedge \neg(\forall x\ P(x) \Rightarrow P{*}(x)) \wedge \Delta(P{*}))$$

Although the use of a second-order formula is unsettling (since we have not described inference techniques for second-order logics), we shall see that, in many important cases, this formula can be reduced to an equivalent first-order formula.

Before discussing methods for simplifying the second-order formula for circumscription, we shall first rewrite it in some alternative forms.

Distributing the negation over the three conjuncts in the circumscription formula and then rewriting the resulting disjunction as an implication yields the usual form for circumscription:

$$\mathrm{CIRC}[\Delta; P] \equiv \Delta \wedge (\forall P{*}\ (\Delta(P{*}) \wedge (\forall x\ P{*}(x) \Rightarrow P(x))) \Rightarrow (\forall x\ P(x) \Rightarrow P{*}(x)))$$

We can get an additional perspective by deriving another form of circumscription. Since the preceding circumscription formula is universally quantified over P*, it holds in particular for $P \wedge P'$ substituted for P* (where P' is a relation constant of the same arity as P):

$$\Delta(P \wedge P') \wedge (\forall x\ P(x) \wedge P'(x) \Rightarrow P(x)) \Rightarrow (\forall x\ P(x) \Rightarrow P(x) \wedge P'(x))$$

This formula reduces to

CIRC formula

$$\Delta(P \wedge P') \Rightarrow (\forall x\ P(x) \Rightarrow P'(x))$$

Since P' is arbitrary, this formula says that P is circumscribed if and only if any alleged strengthening of P (say to $P \wedge P'$) that still satisfies Δ is no real strengthening because P would then already imply P' anyway.

It is convenient to abbreviate $(\forall \mathtt{x}\ P^*(\mathtt{x}) \Rightarrow P(\mathtt{x}))$ by the expression $P^* \leq P$. We also use the abbreviations $(P^* < P)$ for $((P^* \leq P) \wedge \neg(P \leq P^*))$ and $(P^* = P)$ for $((P^* \leq P) \wedge (P \leq P^*))$. These abbreviations also help us to remember that, when $(\forall \mathtt{x}\ P^*(\mathtt{x}) \Rightarrow P(\mathtt{x}))$, an extension corresponding to P^* is a subset of an extension corresponding to P.

In terms of these abbreviations, we can write the circumscription formula as

$$\forall \mathtt{P*}\ \ (\Delta(\mathtt{P*})\ \wedge\ (\mathtt{P*} \leq P))\ \Rightarrow\ (P \leq \mathtt{P*})$$

which is equivalent to

$$\forall \mathtt{P*}\ \ \Delta(\mathtt{P*})\ \Rightarrow\ \neg(\mathtt{P*} < P)$$

or

$$\neg(\exists \mathtt{P*}\ \ \Delta(\mathtt{P*})\ \wedge\ (\mathtt{P*} < P))$$

This last form of the circumscription formula makes the intuitively helpful statement that there can be no P* that, when substituted for P in Δ, still satisfies Δ and that has a corresponding extension that is a strict subset of the extension corresponding to P.

There are several cases in which circumscription can be simplified. The following theorem often is useful:

THEOREM 6.4 *Given a predicate P, an arbitrary belief set $\Delta(P)$ (containing the predicate P), and any P' of the same arity as P but not defined in terms of P, then, if $\Delta(P) \models \Delta(P') \wedge (P' \leq P)$,* $\mathrm{CIRC}[\Delta; P] \equiv \Delta(P) \wedge (P = P')$.

We first discuss the importance of this theorem and then give its proof and an example of its use. The theorem states that, if we have some predicate P' of the same arity as P but not containing P, and if we can prove $\Delta(P') \wedge (P' \leq P)$, given Δ, then $(P = P')$ is equivalent to the circumscription formula for P in Δ. The theorem is most often used to confirm conjectures about circumscription formulas. P' may contain bound predicate variables, so the circumscription formula may still be a second-order formula; in many cases of interest, however, it will be a first-order one.

Proof Assume the conditions of the theorem; namely,

$$\Delta[P] \models \Delta[P'] \wedge (P' \leq P)$$

Left to right: Assume CIRC$[\Delta; P]$; that is, assume

$$\Delta(P) \ \wedge \ (\forall \mathtt{P*} \ \Delta[\mathtt{P*}] \ \wedge \ (\mathtt{P*} \leq P) \ \Rightarrow \ (P \leq \mathtt{P*}))$$

Using the condition of the theorem, we then have

$$\Delta(P') \ \wedge \ (P' \leq P)$$

Universal specialization on the circumscription formula yields

$$\Delta(P') \ \wedge \ (P' \leq P) \ \Rightarrow \ (P \leq P')$$

Modus ponens applied to these last two expressions yields

$$(P \leq P')$$

This result, together with $(P' \leq P)$, yields $(P = P')$.

Right to left: If the circumscription formula did not follow from the conditions of the theorem, we would have some $P*$ such that $\Delta(P*) \wedge (P* < P)$. Assuming $P = P'$ (the right side of the equivalence in the theorem), we then have $\Delta(P*) \wedge (P* < P')$. The conditions of the theorem, however, state that $\Delta(P*)$ logically entails $(P' \leq P*)$—a contradiction. □

As an example of how this theorem is used, let Δ be $\mathtt{P(A)} \wedge (\forall \mathtt{x}\ \mathtt{Q(x)} \Rightarrow \mathtt{P(x)})$. If we were going to perform predicate completion on these clauses, we would rewrite Δ as $(\forall \mathtt{x}\ \mathtt{Q(x)} \vee (\mathtt{x=A}) \Rightarrow \mathtt{P(x)})$. Predicate completion would yield the completion formula $(\forall \mathtt{x}\ \mathtt{P(x)} \Rightarrow \mathtt{Q(x)} \vee (\mathtt{x=A}))$. Since predicate completion was motivated as a technique for minimizing the objects that satisfy a predicate, we might suspect that it gives the same result as does circumscription. We can use Theorem 6.4 to show that it does in this case.

We let the P' of the theorem be the consequent of the completion formula $\mathtt{Q(x)} \vee (\mathtt{x=A})$. Strictly speaking, we should write P' as $(\lambda \mathtt{x}\ \mathtt{Q(x)} \vee (\mathtt{x=A}))$, a *lambda* expression. Now, to use the theorem, we must prove that Δ logically entails $\Delta(P') \wedge (P' \leq P)$.

Substituting $(\lambda \mathtt{x}\ \mathtt{Q(x)} \vee (\mathtt{x=A}))$ for P in Δ yields

$$\Delta(P') \equiv (\forall \mathtt{x}\ \mathtt{Q(x)} \ \Rightarrow \ \mathtt{Q(x)} \vee \mathtt{x=A}) \ \wedge \ (\mathtt{Q(A)} \vee \mathtt{A=A})$$

We see that $\Delta(P')$ is trivially valid. It remains to show that Δ logically implies $(P' \leq P)$; that is, (∀x Q(x) ∨ (x=A)⇒P(x)). The latter formula, however, is just the normal form of Δ. Thus, the conditions of the theorem are satisfied, and the theorem confirms that CIRC[$\Delta; P$] is (∀x Q(x) ∨ (x=A) ⇔ P(x)).

This example can be generalized to show that predicate completion gives the same results as circumscription when Δ consists of clauses solitary in P.

In many cases of interest in AI, CIRC[$\Delta; P$] "collapses" to a first-order formula. (We give some examples later in which it does not.) The simplest case in which circumscription is collapsible is when all occurrences of P in Δ are *positive*. (A formula has a *positive occurrence of* P if P occurs positively in its clause form; it has a *negative occurrence of* P if P occurs negatively in its clause form.)

As an example, consider the case in which Δ is (∃y P(y)). P occurs only positively in Δ. By manipulating expressions in second-order logic, it can be shown that the circumscription of P in Δ is (∃y∀x (x=y) ⇔ P(x)). Circumscribing P in this case limits the extension of P to a minimal nonempty set; i.e., a singleton.

An important case in which circumscription is collapsible can best be thought of as a simple generalization of the solitary condition used in defining predicate completion. Earlier, we defined what is meant for clauses to be *solitary* in a predicate P. Recall that a clause is solitary in P if, whenever it contains a positive occurrence of P, it contains only one occurrence of P. Generalizing this definition, we say that a *formula* is *solitary in* P if and only if it can be written in the following *normal form*:

$$N[P] \wedge (E \leq P)$$

where $N[P]$ is a formula that contains no positive occurrences of P, E is a formula that contains no occurrences of P, and $E \leq P$ is our usual abbreviation for (∀x E(x)⇒P(x)) (where x might be a tuple of variables).

Note that the *normal form* of a conjunction of clauses solitary in P is in the form $E \leq P$. Thus, solitary clauses are a special case of solitary formulas.

For solitary formulas in general, we have the following theorem.

THEOREM 6.5 CIRC[$N[P] \wedge (E \leq P); P$] $\equiv N[E] \wedge (E = P)$, *where* $N[E]$ *is* $N[P]$ *with each occurrence of* P *replaced by* E.

Proof This theorem follows directly from Theorem 6.4. First note that, since $N[P]$ has no positive occurrences of P, $N[P] \wedge (E \leq P)$ logically entails $N[E]$. (This entailment can be thought of as a kind of "generalized resolution.") Thus, the conditions of Theorem 6.4 are satisfied. □

Thus, for solitary formulas, circumscription is collapsible to a first-order formula. We see that circumscription gives the same result as predicate completion does in the special case of clauses solitary in P. Using Theorem 6.5, we can compute circumscription for theories not in clause form, as long as they can be written in normal form.

We illustrate with an example. Let Δ be given by

$$\exists x\ \neg \mathtt{On}(A,x) \wedge \mathtt{On}(A,B)$$

We want to compute the circumscription of On in Δ. We can write Δ in normal form to show that it is solitary in On:

$$(\exists x\ \neg \mathtt{On}(A,x)) \wedge (\forall x \forall y\ x{=}A \wedge y{=}B \Rightarrow \mathtt{On}(x,y))$$

We identify the first conjunct of this expression as $N[\mathtt{On}]$ (it has no positive occurrences of On), and the second as $(E \leq \mathtt{On})$ with $E(\mathtt{x},\mathtt{y}) \equiv (\mathtt{x}{=}\mathtt{A}) \wedge (\mathtt{y}{=}\mathtt{B})$. ($E$ has no occurrences of On.) Thus, by the theorem, CIRC[Δ; On] is

$$(\forall x \forall y\ \mathtt{On}(x,y) \Leftrightarrow x{=}A \wedge y{=}B) \wedge (\exists x\ \neg(x{=}B))$$

(The only thing "on" something is the object denoted by A, and it is on the object denoted by B, and there is at least one object that is not the same as the one denoted by B.)

Some interesting complications arise when attempting circumscription in formulas slightly more general than solitary ones are. Consider an example in which Δ is

$$\mathtt{Ostrich(x)} \Rightarrow \mathtt{Bird(x)}$$

$$\mathtt{Bird(Tweety)} \vee \mathtt{Bird(Sam)}$$

We cannot use Theorem 6.5 to compute the circumscription of Bird in Δ, because Δ is not solitary in Bird. Before we attempt circumscription, however, let us see whether we can guess what sort of augmenting statement circumscription might make about Bird. From Δ, we can guess that there might be two alternative minimizations of Bird; namely,

- $\forall x\ \mathtt{Bird(x)} \Leftrightarrow \mathtt{Ostrich(x)} \vee x{=}\mathtt{Tweety}$
- $\forall x\ \mathtt{Bird(x)} \Leftrightarrow \mathtt{Ostrich(x)} \vee x{=}\mathtt{Sam}$

The belief set is not sufficiently "definite" to allow us to determine which one of these might be appropriate. This indefiniteness makes it impossible to give a single minimizing Bird. Instead, we can merely say something *about* the minimization of Bird; namely, that it has to be one or the other of these two expressions. That is, all we can apparently say about the minimizing Bird is the following:

$$\begin{aligned}&(\forall \mathtt{x}\ \mathtt{Bird(x)} \Leftrightarrow \mathtt{Ostrich(x)} \vee \mathtt{x=Tweety}) \vee\\ &(\forall \mathtt{x}\ \mathtt{Bird(x)} \Leftrightarrow \mathtt{Ostrich(x)} \vee \mathtt{x=Sam})\end{aligned}$$

Indeed, we can use circumscription to derive this formula. The general circumscription formula for Bird in Δ is

$$\begin{aligned}&\forall \mathtt{Bird*}\ \Delta(\mathtt{Bird*}) \wedge (\forall \mathtt{x}\ \mathtt{Bird*(x)} \Rightarrow \mathtt{Bird(x)}) \Rightarrow\\ &\quad(\forall \mathtt{x}\ \mathtt{Bird(x)} \Rightarrow \mathtt{Bird*(x)})\end{aligned}$$

Let us first substitute $\mathtt{Ostrich(x)} \vee (\mathtt{x=Tweety})$ for $\mathtt{Bird*(x)}$. This yields (after reduction)

$$\begin{aligned}&(\forall \mathtt{x}\ \mathtt{Ostrich(x)} \vee \mathtt{x=Tweety} \Rightarrow \mathtt{Bird(x)}) \Rightarrow\\ &\quad(\forall \mathtt{x}\ \mathtt{Bird(x)} \Rightarrow \mathtt{Ostrich(x)} \vee \mathtt{x=Tweety})\end{aligned}$$

Next, we substitute $\mathtt{Ostrich(x)} \vee (\mathtt{x=Sam})$ for $\mathtt{Bird*(x)}$. This yields

$$\begin{aligned}&(\forall \mathtt{x}\ \mathtt{Ostrich(x)} \vee \mathtt{x=Sam} \Rightarrow \mathtt{Bird(x)}) \Rightarrow\\ &\quad(\forall \mathtt{x}\ \mathtt{Bird(x)} \Rightarrow \mathtt{Ostrich(x)} \vee \mathtt{x=Sam})\end{aligned}$$

Neither of these formulas has an antecedent that follows from Δ, but the disjunction of the antecedents does. That is, from Δ, we can prove

$$\begin{aligned}&(\forall \mathtt{x}\ \mathtt{Ostrich(x)} \vee \mathtt{x=Sam} \Rightarrow \mathtt{Bird(x)}) \vee\\ &\quad(\forall \mathtt{x}\ \mathtt{Ostrich(x)} \vee \mathtt{x=Tweety} \Rightarrow \mathtt{Bird(x)})\end{aligned}$$

(In fact, Δ itself can be written in this form. We first rewrite Bird(Tweety) and Bird(Sam) as $(\forall \mathtt{x}\ (\mathtt{x=Tweety}) \Rightarrow \mathtt{Bird(x)})$ and $(\forall \mathtt{x}\ (\mathtt{x=Sam}) \Rightarrow \mathtt{Bird(x)})$, respectively. Then, by distributivity, we write the conjunction of these formulas and $(\forall \mathtt{x}\ \mathtt{Ostrich(x)} \Rightarrow \mathtt{Bird(x)})$ in the form shown previously.)

Since the disjunction of the antecedents of instances of the circumscription formulas follows from Δ, the disjunction of the consequents does also. The disjunction of the consequents, however, is exactly the formula we guessed would be an appropriate statement to make about the minimizing Bird in this case.

The interesting point about this example is that we can derive a slightly more restrictive statement about Bird from the circumscription formula. In this example, Δ does not force us to accept a formula about Bird as general as we have guessed; the perceptive reader may have already noted that the disjunction of definitions can be tighter. The formula we guessed, although true in all Bird-minimal models, does allow some non-Bird-minimal models as well—in particular when *both* Tweety and Sam are birds. We will return to this example after describing how circumscription collapses for a class of formulas more general than solitary ones are.

Next we consider a more general class of formulas—those we call *separable.* A *formula* is *separable* with respect to a predicate P if any one of the following conditions is satisfied:

(1) It has no *positive* occurrences of P.

(2) It is of the form $(\forall \mathbf{x}\ E(\mathbf{x}) \Rightarrow P(\mathbf{x}))$, where $\mathbf{x}$ is a tuple of variables and $E(\mathbf{x})$ is a formula that does not contain P (we again abbreviate to $E \leq P$).

(3) It is composed of conjunctions and disjunctions of separable formulas.

Note that this definition implies that formulas that are solitary with respect to P also are separable with respect to P. Also, as we shall show, all quantifier-free formulas are separable.

Positive occurrences of P in belief sets of this form are separated into isolated components, and this separation makes possible a collapsed version of circumscription—as we shall see.

First, we point out that a rather wide class of formulas can be put in separable form. In the following pairs of equivalent formulas, the one preceded by a dot is a version in which separability (by the above definition) is obvious. (In the first two cases, the formulas also are solitary in P.)

(1) P(A)

- ∀x x=A ⇒ P(x)

(2) ∀y P(F(y))

- ∀x∃y x=F(y) ⇒ P(x)

(3) Bird(Tweety) ∨ Bird(Sam)

- (∀x x=Tweety ⇒ Bird(x)) ∨ (∀x x=Sam ⇒ Bird(x))

(4) <any unquantified formula>

- <move negations in and use the method of example 1. to rewrite any positive occurrences of P>

(5) (∀u P(u,A)) ∨ (∀u P(u,B))

- (∀u∀x x=A ⇒ P(u,x)) ∨ (∀u∀x x=B ⇒ P(u,x))

However, (∀u P(u,A) ∨ P(u,B)) is *not* separable with respect to P because it cannot be written as a propositional combination of separable formulas.

Although our definition of separability can easily be used to check whether or not a formula is separable (taking into account equivalences such as those in the preceding list), it is not obvious what this definition might have to do with circumscription. It happens that there is a *normal form* for separable formulas—somewhat like the normal form used in defining solitary formulas. We describe this normal form next and then show how it is used to compute circumscription.

From the definition of separability, it is straightforward to show that every formula separable with respect to P is equivalent to a formula in the following normal form with respect to P:

$$\bigvee_i [N_i[P] \wedge (E_i \leq P)]$$

where each E_i is a formula that has no occurrences of P, and each $N_i[P]$ is a formula that has no positive occurrences of P.

We obtain this standard form from any conjunction or disjunction of (separable) formulas by distributivity and by applications of the following rules:

$$\begin{aligned}(\phi \Rightarrow P) \ \wedge \ (\psi \Rightarrow P) &\equiv (\phi \vee \psi) \Rightarrow P\\ (\phi \Rightarrow P) \ \vee \ (\psi \Rightarrow P) &\equiv (\phi \wedge \psi) \Rightarrow P\\ (\phi \Rightarrow P) &\equiv \mathtt{T} \wedge (\phi \Rightarrow P)\\ \phi &\equiv \phi \wedge (\mathtt{F} \Rightarrow P)\end{aligned}$$

(The last two rules are sometimes needed to ensure that each disjunct in the normal form has N_i and E_i terms. Use of these rules gives a T for N_i and an F for E_i. In this case, when writing $(E_i \leq P)$ in its expanded form, we write $(\forall \mathtt{x}\ \mathtt{F} \Rightarrow P(\mathtt{x}))$.)

If Δ is in normal form for P, then the circumscription of P in Δ is collapsible to a first-order formula, as defined in the following theorem.

THEOREM 6.6 *Suppose Δ is separable with respect to P and has normal form with respect to P given by*

$$\bigvee_i [N_i[P] \wedge (E_i \leq P)]$$

Then the circumscription of P in Δ is equivalent to

$$\bigvee_i [D_i \wedge (P = E_i)]$$

where D_i is

$$N_i[E_i] \wedge \bigwedge_{j \neq i} \neg[N_j[E_j] \wedge (E_j < E_i)]$$

and each $N[E]$ is $N[P]$ with all occurrences of P replaced by E.

(Recall that the expanded version of $(E_j < E_i)$ is $[(E_j \leq E_i) \land \neg(E_i \leq E_j)]$ which, further expanded, is $(\forall x\ E_j(x) \Rightarrow E_i(x)) \land \neg(\forall x\ E_i(x) \Rightarrow E_j(x))$.)

To demonstrate that circumscription implies a formula of the form $\bigvee_i [N_i[E_i] \land (P = E_i)]$ requires only a simple generalization of the proof of Theorem 6.5. To show that the extra conjuncts can be included in D_i is somewhat more difficult. These extra conjuncts essentially allow us to omit from the disjunction of definitions for P those disjuncts that would be redundant under certain conditions and given the other disjuncts. (The theorem is proved in [Lifschitz 1987b].)

We will illustrate the role of the D_i by some examples later.

In computing circumscription, the result of Theorem 6.6 simplifies in some special cases. If the normal form has only one disjunct, then we have the special case of a formula solitary in P, and D is $N[E]$. Or, if all the N_i are T, then D_i becomes

$$\bigwedge_{j \neq i} (E_i \leq E_j) \lor \neg(E_j \leq E_i)$$

Suppose, for example, that Δ is P(A) ∨ P(B). We rewrite this in normal form for P:

$$(T \land (\forall x\ x{=}A \Rightarrow P(x))) \lor (T \land (\forall x\ x{=}B \Rightarrow P(x)))$$

Here, the normal form has two disjuncts. D_1 and D_2 are, respectively,

$$(\forall x\ x{=}A \Rightarrow x{=}B) \lor (\exists y\ y{=}B \land \neg(y{=}A))$$

and

$$(\forall x\ x{=}B \Rightarrow x{=}A) \lor (\exists y\ y{=}A \land \neg(y{=}B))$$

which are both true, so the circumscription formula is equivalent to

$$(\forall x\ P(x) \Leftrightarrow x{=}A) \lor (\forall x\ P(x) \Leftrightarrow x{=}B)$$

(In computing the D_i for this case, it is helpful to use the equivalence $(\forall x\ (x{=}A) \Rightarrow P(x)) \Leftrightarrow P(A)$.)

In the last example, the D_i "disappeared," and we were left with a simple disjunction of definitions for P. To illustrate how the D_i can act to restrict these disjunctions, consider the following example. Let Δ be given by P(A) ∨ (P(B) ∧ P(C)). In normal form, Δ is

$$(T \land (\forall x\ x{=}A \Rightarrow P(x))) \lor (T \land (\forall x\ x{=}B \lor x{=}C \Rightarrow P(x)))$$

Thus,

$$
\begin{aligned}
N_1 &\equiv N_2 \equiv \mathtt{T} \\
E_1 &\equiv (\lambda\mathtt{x}\ \ \mathtt{x=A}) \\
E_2 &\equiv (\lambda\mathtt{x}\ \ \mathtt{x=B} \lor \mathtt{x=C}) \\
D_1 &\equiv \mathtt{T} \\
D_2 &\equiv \mathtt{A=B=C} \lor (\mathtt{A{\neq}B} \land \mathtt{A{\neq}C})
\end{aligned}
$$

and Theorem 6.6 gives

$$
\begin{aligned}
\mathrm{CIRC}[\Delta;\mathtt{P}] \equiv\ & (\forall\mathtt{x}\ \mathtt{P(x)} \Leftrightarrow \mathtt{x=A}) \lor \\
& ((\forall\mathtt{x}\ \mathtt{P(x)} \Leftrightarrow \mathtt{x=B} \lor \mathtt{x=C}) \land \\
& \ (\mathtt{A=B=C} \lor (\mathtt{A{\neq}B} \land \mathtt{A{\neq}C})))
\end{aligned}
$$

When (A=B=C), the first disjunct suffices, so this formula simplifies to

$$
\begin{aligned}
\mathrm{CIRC}[\Delta;\mathtt{P}] \equiv\ & (\forall\mathtt{x}\ \mathtt{P(x)} \Leftrightarrow \mathtt{x=A}) \lor \\
& ((\forall\mathtt{x}\ \mathtt{P(x)} \Leftrightarrow \mathtt{x=B} \lor \mathtt{x=C}) \land (\mathtt{A{\neq}B} \land \mathtt{A{\neq}C}))
\end{aligned}
$$

This example nicely illustrates the role of the D_i. In this case, they tighten the definitions by taking into consideration the possibility that A may be equal to one of B,C. (If A were equal to one of B,C, then $\Delta \equiv$ P(A), and circumscription would yield simply $(\forall\mathtt{x}\ \mathtt{P(x)} \Leftrightarrow (\mathtt{x=A}))$.)

Let us now reconsider the example that we discussed earlier when we attempted to guess the result of circumscription. In that example, Δ was given by

$$(\forall\mathtt{x}\ \mathtt{Ostrich(x)} \Rightarrow \mathtt{Bird(x)}) \land (\mathtt{Bird(Tweety)} \lor \mathtt{Bird(Sam)})$$

The normal form is

$$
\begin{aligned}
&(\mathtt{T} \land (\forall\mathtt{x}\ \mathtt{Ostrich(x)} \lor \mathtt{x=Tweety} \Rightarrow \mathtt{Bird(x)})) \lor \\
&(\mathtt{T} \land (\forall\mathtt{x}\ \mathtt{Ostrich(x)} \lor \mathtt{x=Sam} \Rightarrow \mathtt{Bird(x)}))
\end{aligned}
$$

Here again, the D_i do not disappear. After some manipulation, we derive

$$D_1 \equiv \mathtt{Sam=Tweety} \lor \neg\mathtt{Ostrich(Sam)} \lor \mathtt{Ostrich(Tweety)}$$

Which, after the making the UNA, becomes

$$\neg\mathtt{Ostrich(Sam)} \lor \mathtt{Ostrich(Tweety)}$$

and

$$D_2 \equiv \mathtt{Tweety=Sam} \lor \neg\mathtt{Ostrich(Tweety)} \lor \mathtt{Ostrich(Sam)}$$

which, again after the UNA is

$$\neg\texttt{Ostrich(Tweety)} \vee \texttt{Ostrich(Sam)}$$

Using these values in Theorem 6.6 yields

$$\begin{aligned}\mathrm{CIRC}[\Delta;\texttt{Bird}] \equiv\ & ((\forall x\ \texttt{Bird}(x) \Leftrightarrow \texttt{Ostrich}(x) \vee x=\texttt{Tweety}) \wedge \\ & (\neg\texttt{Ostrich(Sam)} \vee \texttt{Ostrich(Tweety)})) \vee \\ & ((\forall x\ \texttt{Bird}(x) \Leftrightarrow \texttt{Ostrich}(x) \vee x=\texttt{Sam}) \wedge \\ & (\neg\texttt{Ostrich(Tweety)} \vee \texttt{Ostrich(Sam)}))\end{aligned}$$

The circumscription is more restrictive than is the formula we had guessed earlier. Circumscription says that there are two alternative "minimal definitions" of Bird; namely, either something is a bird if it is an ostrich or Tweety (and that definition need be a possibility only if Sam is not an ostrich or if Tweety is an ostrich), or something is a bird if it is an ostrich or Sam (and that definition need be a possibility only if Tweety is not an ostrich or if Sam is an ostrich). In our guess earlier, we did not narrow our definition as much as we might have; e.g., in the case in which Sam was an ostrich and Tweety was not. In that case, a minimal definition of Bird does not really need to include the possibility of conferring "birdhood" on Tweety (in order to satisfy Δ) because Bird(Tweety) ∨ Bird(Sam) is satisfied by virtue of Sam being an ostrich.

In all the cases considered so far, we were able to construct a first-order formula the addition of which to Δ achieved the effect of circumscribing a predicate in Δ. There are cases, however, in which circumcription does not collapse to a first-order formula. Here is an example: Suppose Δ contains the single formula

$$(\forall u \forall v\ Q(u,v) \Rightarrow P(u,v)) \wedge (\forall u \forall v \forall w\ P(u,v) \wedge P(v,w) \Rightarrow P(u,w))$$

In this case, the difficulty in saying that Δ expresses all and the only information about P is that Δ says quite a bit about P. It says that P is (at least) the transitive closure of Q. To circumscribe P in Δ would require saying that P is identically the transitive closure of Q, and this cannot be expressed in a first-order formula. One way of saying this, of course, is by the circumscription formula itself:

$$\begin{aligned}(\forall P*)\ & (\forall u \forall v\ Q(u,v) \Rightarrow P*(u,v)) \\ & \wedge(\forall u \forall v \forall w\ P*(u,v) \wedge P*(v,w) \Rightarrow P*(u,w)) \\ & \wedge(\forall u \forall v\ P*(u,v) \Rightarrow P(u,v)) \\ & \Rightarrow(\forall u \forall v\ P(u,v) \Rightarrow P*(u,v))\end{aligned}$$

Besides involving a second-order quantifier, this formula is not in the form of a definition for P. We can use Theorem 6.4 to rewrite the circumscription formula in its equivalent definitional form. We leave it to the reader to verify that the following expression for P′ satisfies the conditions of Theorem 6.4:

$$\begin{aligned}&\mathtt{P}'(\mathtt{x},\mathtt{y}) \Leftrightarrow (\forall \mathtt{P}{*}\ (\forall \mathtt{u}\forall \mathtt{v}\ \mathtt{Q}(\mathtt{u},\mathtt{v}) \Rightarrow \mathtt{P}{*}(\mathtt{u},\mathtt{v}))\\ &\quad \wedge(\forall \mathtt{u}\forall \mathtt{v}\forall \mathtt{w}\ (\mathtt{P}{*}(\mathtt{u},\mathtt{v}) \wedge \mathtt{P}{*}(\mathtt{v},\mathtt{w}) \Rightarrow \mathtt{P}{*}(\mathtt{u},\mathtt{w})) \Rightarrow \mathtt{P}{*}(\mathtt{x},\mathtt{y}))\end{aligned}$$

Theorem 6.4 then states that circumscription is equivalent to the following definition for P:

$$\forall \mathtt{u}\forall \mathtt{v}\ \mathtt{P}(\mathtt{u},\mathtt{v}) \Leftrightarrow \mathtt{P}'(\mathtt{u},\mathtt{v})$$

Another example of the insufficiency of a first-order formula to express circumscription comes from the algebraic axioms for natural numbers. Suppose Δ is

$$\mathtt{NN}(0) \wedge (\forall \mathtt{x}\ \mathtt{NN}(\mathtt{x}) \Rightarrow \mathtt{NN}(\mathtt{S}(\mathtt{x})))$$

That is, 0 is a nonnegative integer and the successor of every nonnegative integer is a nonnegative integer. Circumscribing NN in Δ produces an expression equivalent to the usual second-order formula for induction:

$$\begin{aligned}&\forall \mathtt{NN}{*}\ (\mathtt{NN}{*}(0) \wedge (\forall \mathtt{x}\ \mathtt{NN}{*}(\mathtt{x}) \Rightarrow \mathtt{NN}{*}(\mathtt{S}(\mathtt{x}))))\\ &\quad \wedge (\forall \mathtt{x}\ \mathtt{NN}{*}(\mathtt{x}) \Rightarrow \mathtt{NN}(\mathtt{x}))\\ &\quad \Rightarrow (\forall \mathtt{x}\ \mathtt{NN}(\mathtt{x}) \Rightarrow \mathtt{NN}{*}(\mathtt{x}))\end{aligned}$$

Replacing NN*(x) in this expression by [NN′(x) ∧ NN(x)] allows us to write

$$\begin{aligned}&\forall \mathtt{NN}'\ \mathtt{NN}'(0) \wedge (\forall \mathtt{x}\ \mathtt{NN}'(\mathtt{x}) \Rightarrow \mathtt{NN}'(\mathtt{S}(\mathtt{x})))\\ &\quad \Rightarrow (\forall \mathtt{x}\ \mathtt{NN}(\mathtt{x}) \Rightarrow \mathtt{NN}'(\mathtt{x}))\end{aligned}$$

which is closer to the usual induction formula.

The preceding two examples had belief sets that were neither positive nor separable in the predicates being circumscribed, and therefore it is not surprising that circumscription was not collapsible to first-order formulas in those cases.

It is also possible for Δ to have no minimal models. Consider the following set of formulas:

$$\exists x\ NN(x) \wedge (\forall y\ NN(y) \Rightarrow \neg(x=S(y)))$$

$$\forall x\ NN(x) \Rightarrow NN(S(x))$$

$$\forall x \forall y\ S(x)=S(y) \Rightarrow x=y$$

One interpretation of these formulas involves the claims that there is a number that is not the successor of any number, that every number has a successor that is a number, and that two numbers are equal if their successors are equal. One interpretation for NN is that any integer greater than k satisfies NN. A "smaller" interpretation is that any integer greater than $k+1$ satisfies NN—and so on. Thus, there is no NN-minimal model for Δ. Since there is no NN-minimal model, we might expect that the circumscription of these formulas in NN is inconsistent, and indeed it is. (If the circumscription formula had a model, that model would be a minimal model of the formulas.)

Various sufficient conditions have been established for the circumscription of a consistent belief set to be consistent. We state some of the results here without proof.

THEOREM 6.7 *If a belief set Δ is consistent and universal, then the circumscription of P in Δ is consistent. (A set of formulas is said to be* universal *if either it is a set of clauses or if the conjunctive normal form of each of its formulas contains no Skolem functions.)*

THEOREM 6.8 *If a belief set Δ is consistent and separable with respect to P, then the circumscription of P in Δ is consistent.*

Since sets of clauses are universal and since circumscription of P reduces to predicate completion of P in clauses solitary (and thus separable) in P, Theorem 6.2 follows from either Theorem 6.7 or Theorem 6.8. (Theorem 6.3 follows from versions of these theorems extended to a more general case of circumscription, discussed in Section 6.7.)

Theorems 6.7 and 6.8 apply to two different kinds of formulas; namely, universal and separable ones. These two classes are instances of a common general class—the *almost universal formulas.* A formula is *almost universal with respect to P* if it has the form $(\forall \mathbf{x})\varphi$, where $\mathbf{x}$ is a tuple of object variables, and φ does not contain positive occurrences of P in the scope of quantifiers. Every universal formula is, of course, almost universal with respect to any P. It is not difficult to show that any formula that is separable with respect to P also is almost universal with respect to P.

Theorems 6.7 and 6.8 are thus each a special case of Theorem 6.9.

THEOREM 6.9 *If a belief set Δ is consistent and almost universal with respect to P, then the circumscription of P in Δ is consistent.*

6.5 More General Forms of Circumscription

There are more general versions of circumscription that produce stronger results. First, instead of minimizing only a single predicate, we can minimize a set of predicates. The *parallel circumscription* of $\{P_1, P_2, \ldots, P_N\}$ in Δ is given by the same formula as before, except that we now let P stand for a tuple of predicates:

$$\mathrm{CIRC}[\Delta; P] \equiv \Delta(P) \;\wedge\; \neg(\exists \mathtt{P*}\;\; \Delta(\mathtt{P*}) \;\wedge\; (\mathtt{P*} < P))$$

where $\mathtt{P*}$ is a tuple of predicate variables of the same arities as P, $(\mathtt{P*} < P)$ is an abbreviation for $(\mathtt{P*} \leq P) \wedge \neg(P \leq \mathtt{P*})$, and $(\mathtt{P*} \leq P)$ is $(\mathtt{P*}_1 \leq P_1) \wedge \ldots \wedge (\mathtt{P*}_N \leq P_N)$.

Rewriting yields:

$$\mathrm{CIRC}[\Delta; P] \equiv \Delta(P) \;\wedge\; (\forall \mathtt{P*}\;\; (\Delta(\mathtt{P*}) \;\wedge\; (\mathtt{P*} \leq P)) \;\Rightarrow\; (P \leq \mathtt{P*}))$$

Parallel circumscription is not really any more difficult to compute than is ordinary, single-predicate circumscription. Theorem 6.4, for example, is easily generalized to deal with this case. In fact, when all the predicates in the tuple P occur positively in Δ, we have Theorem 6.10.

THEOREM 6.10 *If all occurrences of $P_1, P_2, \ldots, P_N$ in Δ are positive, then* CIRC$[\Delta; P]$ *is equivalent to*

$$\bigwedge_{i=1}^{N} \mathrm{CIRC}[\Delta; P_i]$$

(This theorem is stated without proof in [Lifschitz 1986c], and is proved in [Lifschitz 1987b].)

As an example, we apply Theorem 6.10 to compute the parallel circumscription of {P1,P2} in $(\forall \mathtt{x}\;\; \mathtt{P1(x)} \vee \mathtt{P2(x)})$. Each of P1 and P2 occur positively in Δ, so the parallel circumscription is just the conjunction of the individual circumscriptions of P1 and P2. Since each of CIRC[Δ;P1] and CIRC[Δ;P2] is $(\forall \mathtt{x}\;\; \mathtt{P1(x)} \Leftrightarrow \neg\mathtt{P2(x)})$, their conjunction is also.

The definitions of formulas solitary or separable in P extend naturally to the case in which P is a tuple of predicates. For example, a formula Δ is *solitary in a tuple of predicates* P if it can be written in the form $N[P] \wedge (E \leq P)$, where $N[P]$ has no positive occurrences of any member of P, and no member of E has any occurrences of any member of P. Theorems 6.5 and 6.6 (interpreting P as a tuple of predicates) can then also be used to compute parallel circumscription.

For parallel circumscription, we can state a stronger result than would be obtained by extending Theorem 6.5 to formulas *solitary* in a tuple of predicates. Generalizing on the definition given in Section 6.2 of *clauses* ordered in P, we say that a *formula is ordered* in $P = \{P_1, P_2, \ldots, P_N\}$ if it can be written in the form:

$$N[P] \wedge (E_1 \leq P_1) \wedge (E_2 \leq P_2) \wedge \ldots \wedge (E_N \leq P_N)$$

where $N[P]$ has no positive occurrences of any of the predicates in P, and each E_i has no occurrences of any of $\{P_i, P_{i+1}, \ldots, P_N\}$ and has no negative occurrences of any of $\{P_1, \ldots, P_{i-1}\}$.

Using this definition, we have the following theorem.

THEOREM 6.11 *Suppose Δ is ordered in P and can be written in the form $N[P] \wedge (E_1 \leq P_1) \wedge (E_2[P_1] \leq P_2) \wedge \ldots \wedge (E_N[P_1, P_2, \ldots, P_{N-1}] \leq P_N)$ (with no positive occurrences of the P_i in N and no occurrences of $P_i, \ldots, P_N$ in E_i).*

Then the parallel circumscription of P in Δ is given by

$$\begin{aligned}\mathrm{CIRC}[\Delta; P] \equiv N[E_1, \ldots, E_N] \wedge (P_1 = E_1) \wedge (P_2 = E_2[E_1]) \wedge \ldots \\ \wedge (P_N = E_N[E_1, \ldots, E_{N-1}])\end{aligned}$$

The proof is analogous to that of Theorem 6.5 and, like it, is based on Theorem 6.4.

Note that parallel predicate completion for *clauses* ordered in P is a special case of parallel circumscription.

Another generalization of circumscription allows other predicates to "vary" in addition to those being minimized. That is, we suppose that the extensions of these variable predicates can be changed during the minimization process to allow the predicates being circumscribed to have extensions even smaller than they would have otherwise. This means that an object will be allowed to satisfy one of the variable predicates (to satisfy Δ), so that it does not have to satisfy a predicate being minimized (to satisfy Δ). Which predicates should be allowed to vary depends on the purpose of the circumscription process; the decision is part of what we call the *circumscription policy.* The usual situation is that we are interested in knowing what is the effect of the circumscription of one predicate, P, (or set of predicates) on some other variable predicate, Z, (or set of predicates). We want circumscription to minimize the number of objects satisfying P, even at the expense of allowing more or different objects to satisfy Z, the variable predicate. After defining circumscription with variable predicates, we will give examples of how this process is used.

Suppose P is a tuple of predicates to be minimized and Z (disjoint from P) is a tuple of predicates. The parallel circumscription of P in $\Delta(P;Z)$, with predicates Z also allowed to vary, is

$$\mathrm{CIRC}[\Delta;P;Z] \equiv \Delta(P;Z) \land \neg(\exists \mathtt{P*} \exists \mathtt{Z*}\ \Delta(\mathtt{P*};\mathtt{Z*}) \land (\mathtt{P*} < P))$$

where $\mathtt{P*}$ and $\mathtt{Z*}$ are tuples of predicate variables (of the same arities as P and Z, respectively), and $\Delta(\mathtt{P*};\mathtt{Z*})$ is the belief set expressed as a single wff with all occurrences of P replaced by $\mathtt{P*}$ and all occurrences of Z replaced by $\mathtt{Z*}$.

Rewriting yields

$$\begin{aligned}&\mathrm{CIRC}[\Delta;P;Z]\\ &\equiv \Delta(P;Z) \land (\forall \mathtt{P*} \forall \mathtt{Z*}\ \ (\Delta(\mathtt{P*};\mathtt{Z*}) \land (\mathtt{P*} \le P)) \ \Rightarrow\ (P \le \mathtt{P*}))\\ &\equiv \Delta(P;Z) \land (\forall \mathtt{P*}\ \ (\exists \mathtt{Z*}\ \ (\Delta(\mathtt{P*};\mathtt{Z*}) \land (\mathtt{P*} \le P)) \ \Rightarrow\ (P \le \mathtt{P*})))\\ &\equiv \Delta(P;Z) \land \mathrm{CIRC}[(\exists \mathtt{Z*}\ \ \Delta(P;\mathtt{Z*}));P]\end{aligned}$$

From this form, we see that the parallel circumscription of P in $\Delta(P;Z)$ with Z allowed to vary during the minimization is the same as ordinary parallel circumscription of P in $(\exists \mathtt{Z*}\ \ \Delta[P;\mathtt{Z*}])$. The main difficulty now is how to handle the second-order quantifier in $(\exists \mathtt{Z*}\ \ \Delta[P;\mathtt{Z*}])$.

This problem can be dealt with if Δ is solitary, separable, or ordered in Z. Recall, for example, that if Δ is solitary in Z, it can be written as $N[Z] \land (E \le Z)$, where $N[Z]$ is a formula that contains no positive occurrences of (any elements of) Z, and E is a formula that contains no occurrences of (any elements of) Z. It is straightforward to show that $(\exists \mathtt{Z*}\ \ N[\mathtt{Z*}] \land (E \le \mathtt{Z*})) \equiv N[E]$, where $N[E]$ is $N[\mathtt{Z*}]$ with E substituted for $\mathtt{Z*}$.

These results establish the following theorem for the case in which Δ is solitary in Z.

THEOREM 6.12

$$\mathrm{CIRC}[N(Z) \land (E \le Z);P;Z] \equiv N(Z) \land (E \le Z) \land \mathrm{CIRC}[N(E);P]$$

where N has no positive occurrences of Z, and E has no occurrences of Z. E, P, and Z can be tuples of predicates.

COROLLARY 6.2

$$\mathrm{CIRC}[E_1 \land (E_2 \leq Z); P; Z] \equiv E_1 \land (E_2 \leq Z) \land \mathrm{CIRC}[E_1; P]$$

where neither E_1 nor E_2 has any occurrences of Z. (That is, in this case varying Z lets us drop the clause $(E_2 \leq Z)$ from Δ when computing the circumscription formula.)

A simple default reasoning example will illustrate the effects on circumscription of letting a predicate vary. Let Δ be

$$\forall \mathtt{x}\ \mathtt{Bird(x)} \land \neg\mathtt{Ab(x)} \Rightarrow \mathtt{Flies(x)}$$

$$\forall \mathtt{x}\ \mathtt{Ostrich(x)} \Rightarrow \mathtt{Ab(x)}$$

Ordinary circumscription of Ab in Δ gives us

$$\begin{aligned}&\mathrm{CIRC}[\Delta; \mathtt{Ab}]\\ &\equiv \Delta \land (\forall \mathtt{x}\ \mathtt{Ab(x)} \Leftrightarrow \mathtt{Ostrich(x)} \lor (\mathtt{Bird(x)} \land \neg\mathtt{Flies(x)}))\end{aligned}$$

(The only abnormal things are either ostriches or birds that do not fly.)

A tighter characterization of Ab can be obtained by letting Flies vary. We see that we can use Corollary 6.2 to write

$$\begin{aligned}\mathrm{CIRC}[\Delta; \mathtt{Ab}; \mathtt{Flies}] &\equiv \Delta \land \mathrm{CIRC}[(\forall \mathtt{x}\ \mathtt{Ostrich(x)} \Rightarrow \mathtt{Ab(x)}); \mathtt{Ab}]\\ &\equiv \Delta \land (\forall \mathtt{x}\ \mathtt{Ab(x)} \Leftrightarrow \mathtt{Ostrich(x)})\end{aligned}$$

(The only abnormal things are ostriches. Allowing Flies to vary lets us rule out the possibility that birds might not fly.)

As a more complex example, consider the taxonomic hierarchy we used earlier to illustrate delimited predicate completion. We repeat the formulas used in that example:

$$\mathtt{Flying\text{-}Ostrich(x)} \Rightarrow \mathtt{Ostrich(x)}$$

$$\mathtt{Flying\text{-}Ostrich(x)} \Rightarrow \mathtt{Ab3(x)}$$

$$\mathtt{Ostrich(x)} \Rightarrow \mathtt{Bird(x)}$$

$$\mathtt{Ostrich(x)} \Rightarrow \mathtt{Ab2(x)}$$

$$\mathtt{Bird(x)} \Rightarrow \mathtt{Thing(x)}$$

$$\mathtt{Bird(x)} \Rightarrow \mathtt{Ab1(x)}$$

$$\mathtt{Thing(Tweety)}$$

$$\mathtt{Ostrich(x)} \land \neg\mathtt{Ab3(x)} \Rightarrow \neg\mathtt{Flies(x)}$$

$$\mathtt{Thing(x)} \wedge \neg\mathtt{Ab1(x)} \Rightarrow \neg\mathtt{Flies(x)}$$

$$\mathtt{Bird(x)} \wedge \neg\mathtt{Ab2(x)} \Rightarrow \mathtt{Flies(x)}$$

Default reasoning can be accomplished by parallel circumscription of all predicates except Flies, but letting Flies vary. We let Flies vary because we are willing to let it take on any value required in the minimization of the other predicates. In being unconcerned about the value of Flies, we can use the entire belief set Δ in the minimization process (rather than just the taxonomic part used in delimited completion) to achieve the default assumptions that we want.

Thus, we circumscribe Δ (as before) in the predicates {Flying-Ostrich, Ostrich, Ab3, Bird, Ab2, Thing, Ab1}, letting Flies vary as well. In applying the procedure for parallel circumscription, we note first that Δ is solitary in Flies. To see this, observe that all the preceding clauses except the last one have no positive occurrences of Flies, and the antecedent of the last clause has no occurrence of Flies. Thus, we can use Theorem 6.12 and substitute Bird(x) ∧ ¬Ab2(x) for Flies(x) in all but the last clause of the preceding set to yield

$$\mathtt{Flying\text{-}Ostrich(x)} \Rightarrow \mathtt{Ostrich(x)}$$

$$\mathtt{Flying\text{-}Ostrich(x)} \Rightarrow \mathtt{Ab3(x)}$$

$$\mathtt{Ostrich(x)} \Rightarrow \mathtt{Bird(x)}$$

$$\mathtt{Ostrich(x)} \Rightarrow \mathtt{Ab2(x)}$$

$$\mathtt{Bird(x)} \Rightarrow \mathtt{Thing(x)}$$

$$\mathtt{Bird(x)} \Rightarrow \mathtt{Ab1(x)}$$

$$\mathtt{Thing(Tweety)}$$

$$\mathtt{Ostrich(x)} \wedge \neg\mathtt{Ab3(x)} \Rightarrow \neg(\mathtt{Bird(x)} \wedge \neg\mathtt{Ab2(x)})$$

$$\mathtt{Thing(x)} \wedge \neg\mathtt{Ab1(x)} \Rightarrow \neg(\mathtt{Bird(x)} \wedge \neg\mathtt{Ab2(x)})$$

The last two clauses are subsumed by the fourth and sixth clauses, so the last two clauses can be eliminated. The circumscription we want can then be obtained by ordinary parallel circumscription of {Flying-Ostrich, Ostrich, Ab3, Bird, Ab2, Thing, Ab1} in the conjunction of the first seven clauses (without any predicates variable).

Since these clauses are ordered in {Flying-Ostrich, Ostrich, Ab3, Bird, Ab2, Thing, Ab1}, we can circumscribe by parallel predicate completion to obtain the following completion clauses (just as before):

1. $\mathtt{Thing(x)} \Rightarrow \mathtt{Bird(x)} \vee \mathtt{x{=}Tweety}$
2. $\mathtt{Bird(x)} \Rightarrow \mathtt{Ostrich(x)}$
3. $\mathtt{Ostrich(x)} \Rightarrow \mathtt{Flying\text{-}Ostrich(x)}$

4. `¬Flying-Ostrich(x)`
5. `Ab1(x) ⇒ Bird(x)`
6. `Ab2(x) ⇒ Ostrich(x)`
7. `Ab3(x) ⇒ Flying-Ostrich(x)`

6.6 Default Theories

We also can deal with the problem of nonmonotonic reasoning by defining a logic that uses nonstandard, nonmonotonic inference rules. Let us call these inference rules *default rules*, and the resulting theories *default theories.*

A default rule is an inference rule used to augment Δ under certain specified conditions, which we will soon describe. If D is a set of such rules, we denote an augmentation of Δ (there may be more than one), with respect to D, by $\mathcal{E}[\Delta, D]$. (As before, augmentations include Δ and are closed under ordinary deduction.) Default rules are written in the form

$$\frac{\alpha(x) : \beta(x)}{\gamma(x)}$$

where x is a tuple of individual constant schema variables, and α, β, and γ are wff schemata. (In running text, we write this rule as $\alpha(x) : \beta(x)/\gamma(x)$.)

The expressions above the line specify conditions on $\mathcal{E}[\Delta, D]$ that, if satisfied, permit (roughly speaking) the inclusion in $\mathcal{E}[\Delta, D]$ of the *consequent* below the line. A default rule is interpreted as follows: If there is an instance, X_0, of x for which the ground instance $\alpha(X_0)$ follows from $\mathcal{E}[\Delta, D]$ and for which $\beta(X_0)$ is consistent with $\mathcal{E}[\Delta, D]$, then $\mathcal{E}[\Delta, D]$ includes $\gamma(X_0)$.

The rules are called *default rules* because they are useful in expressing beliefs about statements that are typically, but not necessarily always, true. For example, we might express the belief that birds typically fly by the default rule `Bird`(x) : `Flies`(x) / `Flies`(x). That is, if x is a bird, and if it is consistent to believe that x can fly, then it can be believed that x can fly (or x can fly "by default.") If Δ contained only the formulas `Bird(Tweety)` and `Ostrich(x)⇒¬Flies(x)`, then $\mathcal{E}[\Delta, D]$ would contain `Flies(Tweety)`. If we added to Δ the formula `Ostrich(Tweety)`, use of the default rule would be *blocked* because `Flies(Tweety)` is not consistent with the new Δ. Thus, default theories are nonmonotonic.

Our description of how default rules augment a theory may have been misleadingly simple because default theories may have more than one default rule, and rules may interact. The precise definition of $\mathcal{E}[\Delta, D]$ in terms of Δ and a set of default rules, D, must take into account the contributions of all of the default rules plus the closure of $\mathcal{E}[\Delta, D]$ under ordinary deduction. As we shall see, these interactions operate such as sometimes to warrant the existence of more than one augmentation.

Something like the CWA with respect to a predicate P can be formulated using a default inference rule as follows:

$$\frac{:\neg P(x)}{\neg P(x)}$$

That is, if it is consistent to believe an instance of $\neg P(x)$, then that instance of $\neg P(x)$ may be believed. There is a difference, however, between the effects of the CWA with respect to P and a default theory with this default. The CWA permits inferring an instance of $\neg P(x)$ if that instance is consistent with Δ. The default rulë permits it only if the instance is consistent with $\mathcal{E}[\Delta, D]$. Since there may be other default rules contributing to $\mathcal{E}[\Delta, D]$, the two techniques can lead to different augmentations.

Most applications of default rules involve a special case in which default rules take the form $\alpha(x) : \gamma(x)/\gamma(x)$. These are called *normal* default rules, and theories using them are called *normal* default theories. The CWA-type default rule we mentioned is an example of a normal default rule.

(It also is possible to define more general default rules. Consider one of the form $\alpha(x) : \beta_1(x), \beta_2(x), \ldots, \beta_n(x)/\gamma(x)$, the interpretation of which is that, if a ground instance $\alpha(X_0)$ follows from $\mathcal{E}[\Delta, D]$ and if *each* of the $\beta_i(X_0)$ is separately consistent with $\mathcal{E}[\Delta, D]$, then $\gamma(X_0)$ is in $\mathcal{E}[\Delta, D]$. This rule is different from one of the form $\alpha(x) : \beta_1(x) \wedge \beta_2(x) \wedge \ldots \wedge \beta_n(x)/\gamma(x)$, because the conjunction may be inconsistent with $\mathcal{E}[\Delta, D]$, whereas each conjunct may be separately consistent.)

Default theories have a number of interesting properties. (Some of these are particular to normal default theories.) We state, without proof, some of the more important properties here, and illustrate them with some examples.

(1) Just as circumscription sometimes does not produce a unique definition for a predicate, a default theory may have more than one augmentation. Consider, for example, the following (normal) default rules:

$$:\neg \mathtt{A}/\neg \mathtt{A}$$

$$:\neg \mathtt{B}/\neg \mathtt{B}$$

Now, if Δ is simply $\{\mathtt{A} \vee \mathtt{B}\}$, then there are two possible augmentations of Δ; namely, $\{\mathtt{A} \vee \mathtt{B},\ \neg \mathtt{A}\}$ and $\{\mathtt{A} \vee \mathtt{B},\ \neg \mathtt{B}\}$. Whereas the CWA, with respect to both A and B, would have produced an inconsistent augmentation, our default rules in this example give us two choices. We may select either one as an appropriate augmentation of our beliefs.

(2) The union of the two augmentations of the preceding example is inconsistent. In fact, we have the following result: If a

normal default theory has distinct augmentations, they are mutually inconsistent.

(3) There are default theories with no augmentations. Consider the default :A/¬A. If Δ is empty, so is $\mathcal{E}[\Delta, D]$. However, see (4).

(4) Every normal default theory has an augmentation.

(5) A default theory can have an inconsistent augmentation if and only if Δ itself is inconsistent. Since one can prove anything from an inconsistent augmentation and since augmentations (like theories) are closed under ordinary deduction, if a default theory has an inconsistent augmentation, this is its only augmentation.

(6) If D and D' are sets of normal default rules such that $D' \subseteq D$, then for any $\mathcal{E}[\Delta, D']$ there is an $\mathcal{E}[\Delta, D]$ such that $\mathcal{E}[\Delta, D'] \subseteq \mathcal{E}[\Delta, D]$. Thus, we say that normal default theories are *semimonotonic.* Adding new normal default rules does not require the withdrawal of beliefs, even though adding new beliefs might.

After having specified a set of default rules, how are we to use them to perform the kind of nonmonotonic reasoning inherent in their definitions? Typically, we must decide whether or not the initial beliefs Δ and the default rules D warrant including some arbitrary formula ϕ among the augmented beliefs. That is, we must determine whether or not there is an augmentation $\mathcal{E}[\Delta, D]$ that includes the formula ϕ.

We limit our definition of default proofs to the case of normal default theories. (The computation of augmentations for nonnormal default theories can be extremely complex; in fact, it is not even known yet what might be an appropriate proof theory for nonnormal defaults.) Informally, a default proof of ϕ given Δ and D is just like an ordinary proof of ϕ from Δ, except that the (normal) default rules can be used as rules of inference. The use of a default rule must, in strict accordance with its definition, make the necessary consistency check before inferring its consequent. This test can be performed at the time of rule application in *forward* proofs. *Backward* proofs might best be done in two passes—first ignoring the consistency checks to find potential chains of inferences, and then performing the consistency checks, in forward order, on the default rules in a chain.

Suppose, for example, that D consists of the following two default rules: Bird(x):Flies(x)/Flies(x) (by default, birds can fly), and FC(x):Bird(x)/Bird(x) (by default, feathered creatures are birds). If Δ contains only the statement FC(Tweety), then there is a default proof of Flies(Tweety). If, however, Δ contains instead the statements Ostrich(Tweety), Ostrich(x) ⇒ ¬Flies(x), and Ostrich(x) ⇒ FC(x), then there is no default proof of Flies(Tweety), because the rule instance Bird(Tweety):Flies(Tweety)/Flies(Tweety) cannot be used consistently.

Because default rules interact in complex ways, one must be careful about how knowledge is expressed. An illustrative instance of the sort of knowledge-representation difficulty that might arise is provided by the fact that default rules can be transitive. Suppose, for example, we have $D = \{\mathtt{D}(x){:}\mathtt{A}(x)/\mathtt{A}(x),\ \mathtt{A}(x){:}\mathtt{E}(x)/\mathtt{E}(x)\}$. We might interpret these expressions as saying: Typically, high-school dropouts are adults; and, typically, adults are employed. One of the consequences of these two rules also could be obtained by the combined rule $\mathtt{D}(x){:}\mathtt{E}(x)/\mathtt{E}(x)$, the corresponding interpretation of which would be: Typically, high-school dropouts are employed. Although we might assent to the first two rules, we would not necessarily want to use the combination.

This unwelcome transitivity can be blocked in one of two ways. We could change the second default rule to the nonnormal rule $\mathtt{A}(x){:}[\neg\mathtt{D}(x) \wedge \mathtt{E}(x)]/\mathtt{E}(x)$. Nonnormal defaults do not enjoy the simple and desirable properties of normal defaults, however. We often can block transitivity by a more careful formulation using only normal defaults. In our example, because, typically, adults are not high-school dropouts, we could use the following normal defaults: $\{\mathtt{D}(x){:}\mathtt{A}(x)/\mathtt{A}(x),\ [\mathtt{A}(x) \wedge \neg\mathtt{D}(x)]{:}\mathtt{E}(x)/\mathtt{E}(x),\ \mathtt{A}(x){:}\neg\mathtt{D}(x)/\neg\mathtt{D}(x)\}$. Now we would not conclude that some particular high-school dropout was employed.

6.7 Bibliographical and Historical Remarks

Almost all interesting applications of AI require some form of nonmonotonic reasoning because the knowledge that AI systems contain about their domains is always subject to change and elaboration, yet it is still necessary for AI systems to reason as best they can with the knowledge they currently have. Reiter gave an excellent summary of nonmonotonic reasoning and its applications in AI [Reiter 1987b]. An important, typical application is to the problem of equipment diagnosis [Reiter 1987a]. McCarthy discussed several applications of one type of nonmonotonic reasoning [McCarthy 1986].

The closed-world assumption (CWA) is an important convention of database design. Reiter [Reiter 1978] was the first to describe and prove several of its properties. Theorem 6.1 is taken from [Shepherdson 1984]. The domain-closure assumption (DCA) and unique-names assumption (UNA) were discussed by Reiter [Reiter 1980b].

The *qualification problem* was discussed by McCarthy [McCarthy 1980]. It often is cited as one of the reasons that a strictly logical approach to AI will not work, and it has motivated much of the work in nonmonotonic reasoning.

Predicate completion of a set of clauses was first described by Clark [Clark 1978]. Parallel predicate completion suggests itself by analogy with parallel circumscription. Taxonomic hierarchies are ubiquitous in AI applications. Several frame-based systems have facilities for property inheritance and default reasoning in these hierarchies [Stefik 1986]. Our use of the Ab predicate in this connection is based on a suggestion by McCarthy [McCarthy 1986].

Circumscription as a method of nonmonotonic reasoning was first proposed by McCarthy [McCarthy 1980]. Our notation follows that of Lifschitz [Lifschitz 1985a]. (The alternative form of circumscription—which says that any alleged strengthening of P by P' is no real strengthening because the circumscription of P already implies P'—was attributed by Minker and Perlis [Minker 1984] to Reiter.) The circumscription formula is a formula in second-order logic. Although our treatment of circumscription in this book is essentially limited to those cases that collapse to first order, the reader may want to look at Enderton's chapter on second-order logic [Enderton 1972].

Theorems 6.4 through 6.6 all were developed by Lifschitz; their proofs are given in [Lifschitz 1987b]; Theorems 6.5 and 6.6 are stated without proof in [Lifschitz 1985a]. That CIRC[Δ; P] is collapsible to a first-order formula if all occurrences of P are positive in Δ follows immediately from results in Lifschitz [Lifschitz 1986c], and also was proved in [Lifschitz 1987b].

Etherington, Mercer, and Reiter showed that the circumscription of a formula having no minimal model is inconsistent, and they also proved a sufficient condition on the consistency of circumscription (Theorem 6.7) [Etherington 1985]. Theorem 6.8 was developed by Lifschitz [Lifschitz 1986b]. Theorems 6.7 and 6.8 are each special cases of Theorem 6.9, also developed by Lifschitz [Lifschitz 1986b]. Perlis and Minker [Perlis 1986] also have done work relating properties of circumscription to minimal models.

Parallel circumscription is a natural extension of ordinary circumscription. Theorem 6.10, developed by Lifschitz [Lifschitz 1986c, Lifschitz 1987b], often is useful in computing parallel circumscription. Otherwise, for *ordered* formulas, parallel circumscription can be computed using Theorem 6.11. (Ordered formulas and Theorem 6.11 are original here.) Theorem 6.12, developed by Lifschitz [Lifschitz 1987b], is useful for computing circumscription with variable predicates.

Etherington [Etherington 1986] and Lifschitz [Lifschitz 1986b] independently extended Theorem 6.7 to the case of parallel circumscription with variable predicates. That is, the parallel circumscription of universal theories (even with variable predicates) is consistent if the theory itself is consistent.

Several authors have been concerned with the relationship between circumscription and other nonmonotonic reasoning methods. For example, there are conditions under which parallel circumscription and the CWA augment a belief set identically. Lifschitz [Lifschitz 1985b] showed that the CWA applied to the belief set produces the same effect as does the parallel circumscription of all predicates in the belief set if (1) the CWA can be applied to a belief set consistently, (2) constant terms in the belief set name all possible objects in the domain (the DCA), and (3) different constant terms in the belief set denote different objects in the domain (the UNA). Gelfond, Przymusinska, and Przymusinski studied the relationships among various generalizations of the CWA and circumscription [Gelfond 1986]. Reiter was the first person to show that predicate completion is a special case of circumscription (using an argument similar to that used in the proof of Theorem 6.4) [Reiter 1982].

Przymusinski [Przymusinski 1986] proposed a method for deciding whether or not there is a minimal model of a theory $\mathcal{T}$ that also satisfies a formula ϕ; it can be used to answer queries in circumscriptive theories.

Imielinski and Grosof each investigated relationships between default logics and circumscription [Imielinski 1985, Grosof 1984].

Default logic was originally proposed and analyzed in a paper by Reiter [Reiter 1980a]. Our discussion of default theories is based on that paper. He shows that default logic is not even semidecidable, but describes a resolution theorem prover that can be used for top-down or backward searches for default proofs. Reiter and Criscoulo [Reiter 1983] gave examples of default-rule formulations of typical commonsense reasoning problems and showed how to avoid various pitfalls without using nonnormal defaults.

Other methods of nonmonotonic reasoning also have been proposed. McDermott and Doyle [McDermott 1980, McDermott 1982a] defined a logic having a *modal operator* **M**. (Modal operators are discussed in Chapter 9.) In the semantics for such a logic, the formula $\mathbf{M}P$ has value true just in case P is consistent (with the theory based on Δ). Any derivations of $\mathbf{M}P$ or its consequences are nonmonotonic because the meaning of **M** depends globally on the theory. If we add another formula to Δ, $\mathbf{M}P$ may no longer be consistent. With a somewhat different application in mind, Moore [Moore 1985b] proposed a variant, which he called *autoepistemic logic*, and compared it with McDermott and Doyle's nonmonotonic logic. Konolige [Konolige 1987] analyzed the connections between default theories and autoepistemic logic.

Several additional papers appear in the proceedings of a workshop on nonmonotonic reasoning [Nonmonotonic 1984].

Exercises

1. *Idempotence.* We denote the CWA augmentation of Δ by CWA[Δ]. Prove that

$$\text{CWA}[\text{CWA}[\Delta]] = \text{CWA}[\Delta]$$

(Assume CWA[Δ] is consistent.)

2. *Insensitivity to negative clauses.* Suppose Δ is Horn and consistent. Show that deleting a negative clause (i.e., one without any positive literals) from Δ does not change the CWA augmentation of Δ.

3. *Inconsistencies.* Prove that, if a consistent Δ contains only Horn clauses and is inconsistent with $\neg L_1 \wedge \neg L_2$ (where L_1 and L_2 are positive literals), then either $\Delta \wedge \neg L_1$ or $\Delta \wedge \neg L_2$ is inconsistent also.

4. *Even and odd.* Compute the completion of EVEN in the conjunction of the following formulas:

$$\forall x\ \mathtt{ODD(x)} \wedge \mathtt{x>0} \Rightarrow \mathtt{EVEN(Succ(x))}$$

$$\forall x\ \mathtt{ODD(x)} \wedge \mathtt{x>0} \Rightarrow \mathtt{EVEN(Pred(x))}$$

5. *Integers.* Compute the completion of INT in INT(0) ∧ (INT(x) ⇒ INT(Succ(x))).

6. *Delimited predicate completion.* Discuss how *delimited* predicate completion might produce an inconsistent augmentation.

7. *Completion.* Compute the completion of P in the following clauses:

$$\mathtt{Q1(x)} \wedge \mathtt{Q2(x)} \Rightarrow \mathtt{P(F(x))}$$

$$\mathtt{Q3(x)} \Rightarrow \mathtt{P(G(x))}$$

8. *Is there a* Q *that is not a* P*?* Express in words what the effect would be of circumscribing Q in P < Q.

9. *Parallel.* Compute CIRC[(∀x Q(x) ⇒ P1(x) ∨ P2(x));P1,P2].

10. *Knights and Knaves.* Let Δ be the conjunction of the following formulas:

$$\forall x\ \mathtt{KNIGHT(x)} \Rightarrow \mathtt{PERSON(x)}$$

$$\forall x\ \mathtt{KNAVE(x)} \Rightarrow \mathtt{PERSON(x)}$$

$$\forall x\ \mathtt{KNAVE(x)} \Rightarrow \mathtt{LIAR(x)}$$

$$\exists x\ \neg\mathtt{LIAR(x)} \wedge \neg\mathtt{KNAVE(x)}$$

$$\mathtt{LIAR(MORK)}$$

$$\mathtt{KNAVE(BORK)}$$

a. Compute CIRC[Δ; LIAR].

b. Compute CIRC[Δ; LIAR, KNAVE].

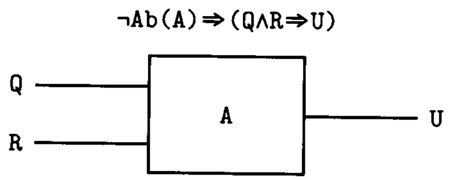

Figure 6.2 An AND gate.

11. *AND gate.* The AND gate, A, shown in Figure 6.2, is described by the following formula:

$$\neg \mathrm{Ab}(A) \Rightarrow (Q \wedge R \Rightarrow U)$$

(Unless A is abnormal, Q and R imply U. Q denotes the proposition "input 1 is on;" R denotes the proposition "input 2 is on;" U denotes the proposition "the output is on.")

a. Suppose, in fact, that Q, R, and U are true. Use the circumscription of Ab in this theory to prove that nothing is abnormal. The "theory" in this case is

$$Q \wedge R \wedge U \wedge (\neg \mathrm{Ab}(A) \Rightarrow (Q \wedge R \Rightarrow U))$$

b. Suppose now, instead, that Q and R are true, but U is false. Use the circumscription of Ab in *this* theory to show that the *only* abnormal thing is A.

12. *Both* P *and* Q. Let Δ consist of these two formulas:

$$\forall x\ R(x) \Rightarrow P(x)$$

$$\forall x\ R(x) \Rightarrow Q(x)$$

Show how to use circumscription to assert that the only objects that satisfy both P and Q also satisfy R. Hint: Use $(\forall x\ (P(x) \wedge Q(x)) \Rightarrow \mathrm{Ab}(x))$ and minimize Ab with P and Q variable.