APPLICATIONS OF CIRCUMSCRIPTION TO FORMALIZING COMMON SENSE KNOWLEDGE

John McCarthy
Computer Science Department
Stanford University
Stanford, CA 94305
jmc@cs.stanford.edu
http://www-formal.stanford.edu/jmc/

1986

Abstract

We present a new and more symmetric version of the circumscription method of nonmonotonic reasoning first described in (McCarthy 1980) and some applications to formalizing common sense knowledge. The applications in this paper are mostly based on minimizing the abnormality of different aspects of various entities. Included are nonmonotonic treatments of *is-a* hierarchies, the unique names hypothesis, and the frame problem. The new circumscription may be called *formula circumscription* to distinguish it from the previously defined *domain circumscription* and *predicate circumscription*. A still more general formalism called *prioritized circumscription* is briefly explored.

1 INTRODUCTION AND NEW DEFINITION OF CIRCUMSCRIPTION

(McCarthy 1980) introduces the circumscription method of nonmonotonic reasoning and gives motivation, some mathematical properties and some examples of its application. The present paper is logically self-contained, but motivation may be enhanced by reading the earlier paper. We don't repeat its arguments about the importance of nonmonotonic reasoning in AI, and its examples are instructive.

Here we give a more symmetric definition of circumscription and applications to the formal expression of common sense facts. Our long term goal (far from realized in the present paper) is to express these facts in a way that would be suitable for inclusion in a general purpose database of common sense knowledge. We imagine this database to be used by AI programs written after the initial preparation of the database. It would be best if the writers of these programs didn't have to be familiar with how the common sense facts about particular phenomena are expressed. Thus common sense knowledge must be represented in a way that is not specific to a particular application.

It turns out that many such common sense facts can be formalized in a uniform way. A single predicate *ab*, standing for "abnormal" is circumscribed with certain other predicates and functions considered as variables that can be constrained to achieve the circumscription subject to the axioms. This also seems to cover the use of circumscription to represent default rules.

2 A NEW VERSION OF CIRCUMSCRIPTION

Definition. Let $A(P)$ be a formula of second order logic, where P is a tuple of some of the free predicate symbols in $A(P)$. Let $E(P,x)$ be a wff in which P and a tuple x of individual variables occur free. The circumscription of $E(P,x)$ relative to $A(P)$ is the formula $A'(P)$ defined by

$$A(P) \wedge \forall P'.[A(P') \wedge [\forall x.E(P',x) \supset E(P,x)] \supset [\forall x.E(P',x) \equiv E(P,x)]]. \quad (1)$$

[We are here writing $A(P)$ instead of $A(P_1,\ldots,P_n)$ for brevity and likewise writing $E(P,x)$ instead of $E(P_1,\ldots,P_n,x_1,\ldots,x_m)$]. Likewise the quantifier $\forall x$ stands for $\forall x_1 \ldots x_m$. $A(P)$ may have embedded quantifiers. Circumscription is a kind of minimization, and the predicate symbols in $A(P)$ that are not in P itself act as parameters in this minimization. When we wish to mention these other predicates we write $A(P\,;Q)$ and $E(P\,;Q,x)$ where Q is a vector of predicate symbols which are not allowed to be varied.

There are two differences between this and (McCarthy 1980). First, in that paper $E(P, x)$ had the specific form $P(x)$. Here we speak of circumscribing a wff and call the method *formula circumscription*, while there we could speak of circumscribing a predicate. We still speak of circumscribing the predicate P when $E(P, x)$ has the special form $P(x)$. Formula circumscription is more symmetric in that any of the predicate symbols in P may be regarded as variables, and a wff is minimized; the earlier form distinguishes one of the predicates themselves for minimization. However, formula circumscription is reducible to predicate circumscription provided we allow as variables predicates besides the one being minimized.

Second, in definition (1) we use an explicit quantifier for the predicate variable P' whereas in (McCarthy 1980), the formula was a schema. One advantage of the present formalism is that now $A'(P)$ is the same kind of formula as $A(P)$ and can be used as part of the axiom for circumscribing some other wff.

In some of the literature, it has been supposed that nonmonotonic reasoning involves giving all predicates their minimum extension. This mistake has led to theorems about what reasoning cannot be done that are irrelevant to AI and database theory, because their premisses are too narrow.

3 A TYPOLOGY OF USES OF NONMONOTONIC REASONING

Before proceeding to applications of circumscription I want to suggest a typology of the uses of nonmonotonic reasoning. Each of the several papers that introduces a mode of nonmonotonic reasoning seems to have a particular application in mind. Perhaps we are looking at different parts of an elephant. The orientation is towards circumscription, but I suppose the considerations apply to other formalisms as well.

Nonmonotonic reasoning has several uses.

1. As a communication convention. Suppose A tells B about a situation involving a bird. If the bird cannot fly, and this is relevant, then A must say so. Whereas if the bird can fly, there is no requirement to mention the fact. For example, if I hire you to build me a bird cage and you don't put a top on it, I can get out of paying for it even if you tell the judge that I never said my bird could fly. However, if I complain that you wasted money by putting

a top on a cage I intended for a penguin, the judge will agree with you that if the bird couldn't fly I should have said so.

The proposed Common Business Communication Language (CBCL) (McCarthy 1982) must include nonmonotonic conventions about what may be inferred when a message leaves out such items as the method of delivery.

2. As a database or information storage convention. It may be a convention of a particular database that certain predicates have their minimal extension. This generalizes the closed world assumption. When a database makes the closed world assumption for all predicates it is reasonable to imbed this fact in the programs that use the database. However, when only some predicates are to be minimized, we need to say which ones by appropriate sentences of the database, perhaps as a preamble to the collection of ground sentences that usually constitute the main content.

Neither 1 nor 2 requires that most birds can fly. Should it happen that most birds that are subject to the communication or about which information is requested from the data base cannot fly, the convention may lead to inefficiency but not incorrectness.

3. As a rule of conjecture. This use was emphasized in (McCarthy 1980). The circumscriptions may be regarded as expressions of some probabilistic notions such as "most birds can fly" or they may be expressions of standard cases. Thus it is simple to conjecture that there are no relevant present material objects other than those whose presence can be inferred. It is also a simple conjecture that a tool asserted to be present is usable for its normal function. Such conjectures sometimes conflict, but there is nothing wrong with having incompatible conjectures on hand. Besides the possibility of deciding that one is correct and the other wrong, it is possible to use one for generating possible exceptions to the other.

4. As a representation of a policy. The example is Doyle's "The meeting will be on Wednesday unless another decision is explicitly made". Again probabilities are not involved.

5. As a very streamlined expression of probabilistic information when numerical probabilities, especially conditional probabilities, are unobtainable. Since circumscription doesn't provide numerical probabilities, its probabilistic interpretation involves probabilities that are either infinitesimal, within an infinitesimal of one, or intermediate — without any discrimination among the intermediate values. The circumscriptions give conditional probabilities. Thus we may treat the probability that a bird can't fly as an infinitesimal. However, if the rare event occurs that the bird is a penguin, then the con-

ditional probability that it can fly is infinitesimal, but we may hear of some rare condition that would allow it to fly after all.

Why don't we use finite probabilities combined by the usual laws? That would be fine if we had the numbers, but circumscription is usable when we can't get the numbers or find their use inconvenient. Note that the general probability that a bird can fly may be irrelevant, because we are interested in the facts that influence our opinion about whether a particular bird can fly in a particular situation.

Moreover, the use of probabilities is normally considered to require the definition of a sample space, i.e. the space of all possibilities. Circumscription allows one to conjecture that the cases we know about are all that there are. However, when additional cases are found, the axioms don't have to be changed. Thus there is no fixed space of all possibilities.

Notice also that circumscription does not provide for weighing evidence; it is appropriate when the information permits snap decisions. However, many cases nominally treated in terms of weighing information are in fact cases in which the weights are such that circumscription and other defaults work better.

6. Auto-epistemic reasoning. "If I had an elder brother, I'd know it". This has been studied by R. Moore. Perhaps it can be handled by circumscription.

7. Both common sense physics and common sense psychology use nonmonotonic rules. An object will continue in a straight line if nothing interferes with it. A person will eat when hungry unless something prevents it. Such rules are open ended about what might prevent the expected behavior, and this is required, because we are always encountering unexpected phenomena that modify the operation of our rules. Science, as distinct from common sense, tries to work with exceptionless rules. However, this means that common sense reasoning has to decide when a scientific model is applicable, i.e. that there are no important phenomena not taken into account by the theories being used and the model of the particular phenomena.

Seven different uses for nonmonotonic reasoning seem too many, so perhaps we can condense later.

4 MINIMIZING ABNORMALITY

Many people have proposed representing facts about what is "normally" the case. One problem is that every object is abnormal in some way, and we want to allow some aspects of the object to be abnormal and still assume the normality of the rest. We do this with a predicate *ab* standing for "abnormal". We circumscribe $ab\ z$. The argument of *ab* will be some aspect of the entities involved. Some aspects can be abnormal without affecting others. The aspects themselves are abstract entities, and their unintuitiveness is somewhat a blemish on the theory.

The idea is illustrated by the examples of the following sections.

5 WHETHER BIRDS CAN FLY

Marvin Minsky challenged us advocates of formal systems based on mathematical logic to express the facts and nonmonotonic reasoning concerning the ability of birds to fly.

There are many ways of nonmonotonically axiomatizing the facts about which birds can fly. The following axioms using *ab* seem to me quite straightforward.

$$\forall x. \neg ab\ aspect1\ x \supset \neg flies\ x. \tag{2}$$

Unless an object is abnormal in $aspect1$, it can't fly. (We're using a convention that parentheses may be omitted for functions and predicates of one argument, so that (2) is the same as $\forall x.(\neg ab(aspect1(x)) \supset \neg flies(x))$.)

It wouldn't work to write $ab\ x$ instead of $ab\ aspect1\ x$, because we don't want a bird that is abnormal with respect to its ability to fly to be automatically abnormal in other respects. Using aspects limits the effects of proofs of abnormality.

$$\forall x. bird\ x \supset ab\ aspect1\ x. \tag{3}$$

$$\forall x. bird\ x \wedge \neg ab\ aspect2\ x \supset flies\ x. \tag{4}$$

Unless a bird is abnormal in $aspect2$, it can fly.

A bird is abnormal in $aspect1$, so (2) can't be used to show it can't fly. If (3) were omitted, when we did the circumscription we would only be able to infer a disjunction. Either a bird is abnormal in $aspect1$ or it can fly unless it is abnormal in $aspect2$. (3) expresses our preference for inferring that a bird is abnormal in $aspect1$ rather than $aspect2$. We call (3) a *cancellation* *of inheritance* axiom.

$$\forall x. ostrich\ x \supset ab\ aspect2\ x. \tag{5}$$

Ostriches are abnormal in *aspect*2. This doesn't say that an ostrich cannot fly — merely that (4) can't be used to infer that it does. (5) is another cancellation of inheritance axiom.

$$\forall x. penguin\ x \supset ab\ aspect2\ x. \tag{6}$$

Penguins are also abnormal in *aspect*2.

$$\forall x. ostrich\ x \wedge \neg ab\ aspect3\ x \supset \neg flies\ x. \tag{7}$$

$$\forall x. penguin\ x \wedge \neg ab\ aspect4\ x \supset \neg flies\ x. \tag{8}$$

Normally ostriches and penguins can't fly. However, there is an out. (7) and (8) provide that under unspecified conditions, an ostrich or penguin might fly after all. If we give no such conditions, we will conclude that an ostrich or penguin can't fly. Additional objects that can fly may be specified. Each needs two axioms. The first says that it is abnormal in *aspect*1 and prevents (2) from being used to say that it can't fly. The second provides that it can fly unless it is abnormal in yet another way. Additional non-flying birds can also be provided for at a cost of two axioms per kind.

We haven't yet said that ostriches and penguins are birds, so let's do that and throw in that canaries are birds also.

$$\forall x. ostrich\ x \supset bird\ x. \tag{9}$$

$$\forall x. penguin\ x \supset bird\ x. \tag{10}$$

$$\forall x. canary\ x \supset bird\ x. \tag{11}$$

Asserting that ostriches, penguins and canaries are birds will help inherit other properties from the class of birds. For example, we have

$$\forall x. bird\ x \wedge \neg ab\ aspect5\ x \supset feathered\ x. \tag{12}$$

So far there is nothing to prevent ostriches, penguins and canaries from overlapping. We could write disjointness axioms like

$$\forall x. \neg ostrich\ x \vee \neg penguin\ x, \tag{13}$$

but we require n^2 of them if we have n species. It is more efficient to write axioms like

$$\forall x. ostrich\ x \supset species\ x = {}'ostrich, \tag{14}$$

which makes the n species disjoint with only n axioms assuming that the distinctness of the names is apparent to the reasoner. This problem is like the unique names problem.

If these are the only facts to be taken into account, we must somehow specify that what can fly is to be determined by circumscribing the wff *ab z* using *ab* and *flies* as variables. Why exactly these? If *ab* were not taken as variable, *ab z* couldn't vary either, and the minimization problem would go away. Since the purpose of the axiom set is to describe what flies, the predicate *flies* must be varied also. Suppose we contemplate taking *bird* as variable also. In the first place, this violates an intuition that deciding what flies follows deciding what is a bird in the common sense situations we want to cover. Secondly, if we use exactly the above axioms and admit bird as a variable, we will further conclude that the only birds are penguins, canaries and ostriches. Namely, for these entities something has to be abnormal, and therefore minimizing *ab z* will involve making as few entities as possible penguins, canaries and ostriches. If we also admit *penguin*, *ostrich*, and *canary* as variable, we will succeed in making *ab z* always false, and there will be no birds at all.

However, if the same circumscriptions are done with additional axioms like *canary Tweety* and *ostrich Joe*, we will get the expected result that Tweety can fly and Joe cannot even if all the above are variable.

While this works it may be more straightforward, and therefore less likely to lead to subsequent trouble, to circumscribe birds, ostriches and penguins with axioms like

$$\forall x. \neg ab\ aspect6\ x \supset \neg bird\ x, \tag{15}$$

We have not yet specified how a program will know what to circumscribe. One extreme is to build it into the program, but this is contrary to the declarative spirit. However, a statement of what to circumscribe isn't just a sentence of the language because of its nonmonotonic character. Another possibility is to include some sort of metamathematical statement like

$$circumscribe(ab\ z\,; ab, flies, bird, ostrich, penguin) \tag{16}$$

in a “policy” database available to the program. (16) is intended to mean that *ab z* is to be circumscribed with *ab*, $flies$, *bird*, *ostrich* and *penguin* taken as variable. Explicitly listing the variables makes adding new kinds awkward, since they will have to be mentioned in the *circumscribe* statement. Section 11 on *simple abnormality theories* presents yet another possibility.

6 THE UNIQUE NAMES HYPOTHESIS

Raymond Reiter (1980b) introduced the phrase “unique names hypothesis” for the assumption that each object has a unique name, i.e. that distinct names denote distinct objects. We want to treat this nonmonotonically. Namely, we want a wff that picks out those models of our initial assumptions that maximize the inequality of the denotations of constant symbols. While we’re at it, we might as well try for something stronger. We want to maximize the extent to which distinct terms designate distinct objects. When there is a unique model of the axioms that maximizes distinctness, we can put it more simply; two terms denote distinct objects unless the axioms force them to denote the same. If we are even more fortunate, as we are in the examples to be given, we can say that two terms denote distinct objects unless their equality is provable.

We don’t know a completely satisfactory way of doing this. Suppose that we have a language L and a theory T consisting of the consequences of a formula A. It would be most pleasant if we could just circumscribe equality, but as Etherington, Mercer and Reiter (1985) point out, this doesn’t work, and nothing similar works. We could hope to circumscribe some other formula of L, but this doesn’t seem to work either. Failing that, we could hope for some other second order formula taken from L that would express the unique names hypothesis, but we don’t presently see how to do it.

Our solution involves extending the language by introducing the names themselves as the only objects. All assertions about objects are expressed as assertions about the names.

We suppose our theory is such that the names themselves are all provably distinct. There are several ways of doing this. Let the names be n_1, n_2, etc.

The simplest solution is to have an axiom $n_i \neq n_j$ for each pair of distinct names. This requires a number of axioms proportional to the square of the number of names, which is sometimes objectionable. The next solution involves introducing an arbitrary ordering on the names. We have special axioms $n_1 < n_2, n_2 < n_3, n_3 < n_4$, etc. and the general axioms $\forall xy.x < y \supset x \neq y$ and $\forall xyz.x < y \wedge y < z \supset x < z$. This makes the number of axioms proportional to the number of names. A third possibility involves mapping the names onto integers with axioms like $index\ n_1 = 1, index\ n_2 = 2$, etc. and using a theory of the integers that provides for their distinctness. The fourth possibility involves using string constants for the names and "attaching" to equality in the language a subroutine that computes whether two strings are equal. If our names were quoted symbols as in LISP, this amounts to having $'a \neq 'b$ and all its countable infinity of analogs as axioms. Each of these devices is useful in appropriate circumstances.

From the point of view of mathematical logic, there is no harm in having an infinity of such axioms. From the computational point of view of a theorem proving or problem solving program, we merely suppose that we rely on the computer to generate the assertion that two names are distinct whenever this is required, since a subroutine can easily tell whether two strings are the same.

Besides axiomatizing the distinctness of the constants, we also want to axiomatize the distinctness of terms. This may be accomplished by providing for each function two axioms. Letting foo be a function of two arguments we postulate

$$\forall x_1 x_2 y_1 y_2 . foo(x_1, y_1) = foo(x_2, y_2) \supset x_1 = x_2 \wedge y_1 = y_2 \tag{17}$$

and

$$\forall xy.fnamefoo(x, y) = 'foo. \tag{18}$$

The first axiom ensures that unless the arguments of foo are identical, its values are distinct. The second ensures that the values of foo are distinct from the values of any other function or any constant, assuming that we refrain from naming any constant $'foo$.

These axioms amount to making our domain isomorphic to an extension of the Herbrand universe of the language.

Now that the names are guaranteed distinct, what about the objects they denote? We introduce a predicate $e(x, y)$ and axiomatize it to be an equivalence relation. Its intended interpretation is that the names x and y denote the same object. We then formulate all our usual axioms in terms of names rather than in terms of objects. Thus $on(n_1, n_2)$ means that the object named by n_1 is on the object named by n_2, and *bird* x means that the name x denotes a bird. We add axioms of substitutivity for e with regard to those predicates and functions that are translates of predicates referring to objects rather than predicates on the names themselves. Thus for a predicate *on* and a function foo we may have axioms

$$\forall n_1 n_2 n'_1 n'_2 . e(n_1, n'_1) \wedge e(n_2, n'_2) \supset (on(n_1, n_2) \equiv on(n'_1, n'_2)) \tag{19}$$

and

$$\forall x_1 x_2 y_1 y_2 . e(x_1, x_2) \wedge e(y_1, y_2) \supset e(foo(x_1, y_1), foo(x_2, y_2)). \tag{20}$$

If for some class C of names, we wish to assert the unique names hypothesis, we simply use an axiom like

$$\forall n_1 n_2 . n_1 \in C \wedge n_2 \in C \supset (e(n_1, n_2) \equiv n_1 = n_2). \tag{21}$$

However, we often want only to assume that distinct names denote distinct objects when this doesn't contradict our other assumptions. In general, our axioms won't permit making all names distinct simultaneously, and there will be several models with maximally distinct objects. The simplest example is obtained by circumscribing $e(x, y)$ while adhering to the axiom

$$e(n_1, n_2) \vee e(n_1, n_3)$$

where n_1, n_2, and n_3 are distinct names. There will then be two models, one satisfying $e(n_1, n_2) \wedge \neg e(n_1, n_3)$ and the other satisfying $\neg e(n_1, n_2) \wedge e(n_1, n_3)$.

Thus circumscribing $e(x, y)$ maximizes uniqueness of names. If we only want unique names for some class C of names, then we circumscribe the formula

$$x \in C \wedge y \in C \supset e(x, y). \tag{22}$$

An example of such a circumscription is given in Appendix B. However, there seems to be a price. Part of the price is admitting names as objects. Another part is admitting the predicate $e(x, y)$ which is substitutive for predicates and functions of names that really are about the objects denoted by the names. $e(x, y)$ is not to be taken as substitutive for predicates on names that aren't about the objects. Of these our only present example is equality. Thus we don't have

$$\forall n_1 n_2 n_1' n_2'.e(n_1, n_1') \wedge e(n_2, n_2') \supset (n_1 = n_2 \equiv n_1' = n_2').$$

The awkward part of the price is that we must refrain from any functions whose values are the objects themselves rather than names. They would spoil the circumscription by not allowing us to infer the distinctness of the objects denoted by distinct names. Actually, we can allow them provided we don't include the axioms involving them in the circumscription. Unfortunately, this spoils the other property of circumscription that lets us take any facts into account.

The examples of the use of circumscription in AI in the rest of the paper don't interpret the variables as merely ranging over names. Therefore, they are incompatible with getting unique names by circumscription as described in this section. Presumably it wouldn't be very difficult to revise those axioms for compatibility with the present approach to unique names.

7 TWO EXAMPLES OF RAYMOND REITER

Reiter asks about representing, "Quakers are normally pacifists and Republicans are normally non-pacifists. How about Nixon, who is both a Quaker and a Republican?" Systems of nonmonotonic reasoning that use non-provability as a basis for inferring negation will infer that Nixon is neither a pacifist nor a non-pacifist. Combining these conclusions with the original premiss leads to a contradiction. We use

$$\forall x.quaker\ x \wedge \neg ab\ aspect1\ x \supset pacifist\ x, \tag{23}$$

$$\forall x.republican\ x \wedge \neg ab\ aspect2\ x \supset \neg pacifist\ x \tag{24}$$

and

$$quaker\ Nixon \wedge republican\ Nixon. \tag{25}$$

When we circumscribe *ab z* using these three sentences as $A(ab, pacifist)$, we will only be able to conclude that Nixon is either abnormal in *aspect*1 or in *aspect*2, and we will not be able to say whether he is a pacifist. Of course, this is the same conclusion as would be reached without circumscription. The point is merely that we avoid contradiction.

Reiter's second example is that a person normally lives in the same city as his wife and in the same city as his employer. But A's wife lives in Vancouver and A's employer is in Toronto. We write

$$\forall x. \neg ab\ aspect1\ x \supset city\ x = city\ wife\ x \tag{26}$$

and

$$\forall x. \neg ab\ aspect2\ x \supset city\ x = city\ employer\ x. \tag{27}$$

If we have

$$city\ wife\ A = Vancouver \wedge city\ employer\ A = Toronto \wedge Toronto \neq Vancouver, \tag{28}$$

we will again only be able to conclude that A lives either in Toronto or Vancouver. In this circumscription, the function *city* must be taken as variable. This might be considered not entirely satisfactory. If one knows that a person either doesn't live in the same city as his wife or doesn't live in the same city as his employer, then there is an increased probability that he doesn't live in the same city as either. A system that did reasoning of this kind would seem to require a larger body of facts and perhaps more explicitly metamathematical reasoning. Not knowing how to do that, we might want to use $aspect1\ x$ in both (26) and (27). Then we would conclude nothing about his city once we knew that he wasn't in the same city as both.

8 A MORE GENERAL TREATMENT OF AN *IS-A* HIERARCHY

The bird example works fine when a fixed *is-a* hierarchy is in question. However, our writing the inheritance cancellation axioms depended on knowing exactly from what higher level the properties were inherited. This doesn't correspond to my intuition of how we humans represent inheritance. It would seem rather that when we say that birds can fly, we don't necessarily have in mind that an inheritance of inability to fly from things in general is being cancelled. We can formulate inheritance of properties in a more general way provided we reify the properties. Presumably there are many ways of doing this, but here's one that seems to work.

The first order variables of our theory range over classes of objects (denoted by c with numerical suffixes), properties (denoted by p) and objects (denoted by x). We don't identify our classes with sets (or with the classes of Gödel-Bernays set theory). In particular, we don't assume extensionality. We have several predicates:

$ordinarily(c, p)$ means that objects of class c ordinarily have property p. $c1 \leq c2$ means that class $c1$ ordinarily inherits from class $c2$. We assume that this relation is transitive. $in(x, c)$ means that the object x is in class c. $ap(p, x)$ means that property p applies to object x. Our axioms are

$$\forall c1c2c3.c1 \leq c2 \wedge c2 \leq c3 \supset c1 \leq c3, \tag{29}$$

$$\forall c1c2p.ordinarily(c2, p) \wedge c1 \leq c2 \wedge \neg ab\, aspect1(c1, c2, p) \supset ordinarily(c1, p), \tag{30}$$

$$\forall c1c2c3p.c1 \leq c2 \wedge c2 \leq c3 \wedge ordinarily(c2, not\ p) \supset ab\ aspect1(c1, c3, p), \tag{31}$$

$$\forall xcp.in(x, c) \wedge ordinarily(c, p) \wedge \neg ab\ aspect2(x, c, p) \supset ap(p, x), \tag{32}$$

$$\forall xc1c2p.in(x, c1) \wedge c1 \leq c2 \wedge ordinarily(c1, not\ p) \supset ab\ aspect2(x, c2, p). \tag{33}$$

Axiom (29) is the afore-mentioned transitivity of $\leq$. (30) says that properties that ordinarily hold for a class are inherited unless something is abnormal. (31) cancels the inheritance if there is an intermediate class for which the property ordinarily doesn't hold. (32) says that properties which ordinarily hold actually hold for elements of the class unless something is abnormal. (33) cancels the effect of (32) when there is an intermediate class for which the negation of the property ordinarily holds. Notice that this reification of properties seems to require imitation boolean operators. Such operators are discussed in (McCarthy 1979).

9 THE BLOCKS WORLD

The following set of "situation calculus" axioms solves the frame problem for a blocks world in which blocks can be moved and painted. Here $result(e, s)$

denotes the situation that results when event e occurs in situation s. The formalism is approximately that of (McCarthy and Hayes 1969).

$$\forall xes.\neg ab\ aspect1(x,e,s) \supset location(x,result(e,s)) = location(x,s). \tag{34}$$

$$\forall xes.\neg ab\ aspect2(x,e,s) \supset color(x,result(e,s)) = color(x,s). \tag{35}$$

Objects change their locations and colors only for a reason.

$$\forall xls.ab\ aspect1(x,move(x,l),s) \tag{36}$$

and

$$\forall xls.\neg ab\ aspect3(x,l,s) \supset location(x,result(move(x,l),s)) = l. \tag{37}$$

$$\forall xcs.ab\ aspect2(x,paint(x,c),s) \tag{38}$$

and

$$\forall xcs.\neg ab\ aspect4(x,c,s) \supset color(x,result(paint(x,c),s)) = c. \tag{39}$$

Objects change their locations when moved and their colors when painted.

$$\forall xls.\neg clear(topx,s) \vee \neg clear(l,s) \vee tooheavy\ x \vee l = top\ x \tag{40}$$

$$\supset ab\ aspect3(x,l,s). \tag{41}$$

This prevents the rule (36) from being used to infer that an object will move if its top isn't clear or to a destination that isn't clear or if the object is too heavy. An object also cannot be moved to its own top.

$$\forall ls.clear(l,s) \equiv \neg\exists x.(\neg trivial\ x \wedge location(x,s) = l). \tag{42}$$

A location is clear if all the objects there are trivial, e.g. a speck of dust.

$$\forall x.\neg ab\ aspect5\ x \supset \neg trivial\ x. \tag{43}$$

Trivial objects are abnormal in *aspect*5.

10 AN EXAMPLE OF DOING THE CIRCUMSCRIPTION

In order to keep the example short we will take into account only the following facts from the earlier section on flying.

$$\forall x.\neg ab\ aspect1\ x \supset \neg flies\ x. \tag{2}$$

$$\forall x.bird\ x \supset ab\ aspect1\ x. \tag{3}$$

$$\forall x.bird\ x \wedge \neg ab\ aspect2\ x \supset flies\ x. \tag{4}$$

$$\forall x.ostrich\ x \supset ab\ aspect2\ x. \tag{5}$$

$$\forall x.ostrich\ x \wedge \neg ab\ aspect3\ x \supset \neg flies\ x. \tag{7}$$

Their conjunction is taken as $A(ab, flies)$. This means that what entities satisfy ab and what entities satisfy $flies$ are to be chosen so as to minimize $ab\ z$. Which objects are birds and ostriches are parameters rather than variables, i.e. what objects are birds is considered given.

We also need an axiom that asserts that the aspects are different. Here is a straightforward version that would be rather long were there more than three aspects.

$$\begin{gathered}(\forall xy.\neg(aspect1\ x = aspect2\ y))\\ \wedge(\forall xy.\neg(aspect1\ x = aspect3\ y))\\ \wedge(\forall xy.\neg(aspect2\ x = aspect3\ y))\\ \wedge(\forall xy.aspect1\ x = aspect1\ y \equiv x = y)\\ \wedge(\forall xy.aspect2\ x = aspect2\ y \equiv x = y)\\ \wedge(\forall xy.aspect3\ x = aspect3\ y \equiv x = y).\end{gathered}$$

We could include this axiom in $A(ab, flies)$, but as we shall see, it won't matter whether we do, because it contains neither ab nor $flies$. The circumscription formula $A'(ab, flies)$ is then

$$A(ab, flies) \wedge \forall ab' flies'.[A(ab', flies') \wedge [\forall x.ab'\ x \supset ab\ x] \supset [\forall x.ab\ x \equiv ab'\ x]], \tag{41}$$

which spelled out becomes

$$\begin{aligned}
&[\forall x.\neg \mathit{ab}\ \mathit{aspect}1\ x \supset \neg \mathit{flies}\ x]\\
&\wedge[\forall x.\mathit{bird}\ x \supset \mathit{ab}\ \mathit{aspect}1\ x]\\
&\wedge[\forall x.\mathit{bird}\ x \wedge \neg \mathit{ab}\ \mathit{aspect}2\ x \supset \mathit{flies}\ x]\\
&\wedge[\forall x.\mathit{ostrich}\ x \supset \mathit{ab}\ \mathit{aspect}2\ x]\\
&\wedge[\forall x.\mathit{ostrich}\ x \wedge \neg \mathit{ab}\ \mathit{aspect}3\ x \supset \neg \mathit{flies}\ x]\\
&\wedge\forall \mathit{ab}'\ \mathit{flies}'.\ [[\forall x.\neg \mathit{ab}'\ \mathit{aspect}1\ x \supset \neg \mathit{flies}'\ x]\\
&\qquad\qquad\qquad \wedge[\forall x.\mathit{bird}\ x \supset \mathit{ab}'\ \mathit{aspect}1\ x]\\
&\qquad\qquad\qquad \wedge[\forall x.\mathit{bird}\ x \wedge \neg \mathit{ab}'\ \mathit{aspect}2\ x \supset \mathit{flies}'\ x]\\
&\qquad\qquad\qquad \wedge[\forall x.\mathit{ostrich}\ x \supset \mathit{ab}'\ \mathit{aspect}2\ x]\\
&\qquad\qquad\qquad \wedge[\forall x.\mathit{ostrich}\ x \wedge \neg \mathit{ab}'\ \mathit{aspect}3\ x \supset \neg \mathit{flies}'\ x]\\
&\qquad\qquad\qquad \wedge[\forall z.\mathit{ab}'\ z \supset \mathit{ab}\ z]\\
&\supset [\forall z.\mathit{ab}\ z \equiv \mathit{ab}'\ z]].
\end{aligned}\tag{42}$$

$A(\mathit{ab}, \mathit{flies})$ is guaranteed to be true, because it is part of what is assumed in our common sense database. Therefore (42) reduces to

$$\begin{aligned}
&\forall \mathit{ab}'\mathit{flies}'.[[\forall x.\neg \mathit{ab}'\ \mathit{aspect}1\ x \supset \neg \mathit{flies}'\ x]\\
&\wedge[\forall x.\mathit{bird}\ x \supset \mathit{ab}'\ \mathit{aspect}1\ x]\\
&\wedge[\forall x.\mathit{bird}\ x \wedge \neg \mathit{ab}'\ \mathit{aspect}2\ x \supset \mathit{flies}'\ x]\\
&\wedge[\forall x.\mathit{ostrich}\ x \supset \mathit{ab}'\ \mathit{aspect}2\ x]\\
&\wedge[\forall x.\mathit{ostrich}\ x \wedge \neg \mathit{ab}'\ \mathit{aspect}3\ x \supset \neg \mathit{flies}'\ x]\\
&\wedge[\forall z.\mathit{ab}'\ z \supset \mathit{ab}\ z]\\
&\supset [\forall z.\mathit{ab}\ z \equiv \mathit{ab}'\ z]].
\end{aligned}\tag{43}$$

Our objective is now to make suitable substitutions for ab' and flies' so that all the terms preceding the $\supset$ in (43) will be true, and the right side will determine ab. The axiom $A(\mathit{ab}, \mathit{flies})$ will then determine flies, i.e. we will know what the fliers are. flies' is easy, because we need only apply wishful thinking; we want the fliers to be just those birds that aren't ostriches. Therefore, we put

$$\mathit{flies}'\ x \equiv \mathit{bird}\ x \wedge \neg \mathit{ostrich}\ x. \tag{44}$$

ab' isn't really much more difficult, but there is a notational problem. We define

$$\mathit{ab}'\ z \equiv [\exists x.\mathit{bird}\ x \wedge z = \mathit{aspect}1\ x] \vee [\exists x.\mathit{ostrich}\ x \wedge z = \mathit{aspect}2\ x], \tag{45}$$

which covers the cases we want to be abnormal.

Appendix A contains a complete proof as accepted by Jussi Ketonen's (1984) interactive theorem prover EKL. EKL uses the theory of types and therefore has no problem with the second order logic required by circumscription.

11 SIMPLE ABNORMALITY THEORIES

The examples in this paper all circumscribe the predicate *ab*. However, they differ in what they take as variable in the circumscription. The declarative expression of common sense requires that we be definite about what is circumscribed and what is variable in the circumscription. We have the following objectives.

1. The general facts of common sense are described by a collection of sentences that are not oriented in advance to particular problems.

2. It is prescribed how to express the facts of particular situations including the goals to be achieved.

3. The general system prescribes what is to be circumscribed and what is variable.

4. Once the facts to be taken into account are chosen, the circumscription process proceeds in a definite way resulting in a definite theory — in general second order.

5. The conclusions reached taking a given set of facts into account are intuitively reasonable.

These objectives are the same as those of (McCarthy 1959) except that that paper used only monotonic reasoning.

The examples of this paper suggest defining a *simple abnormality formalism* used as follows.

1. The general facts include *ab* and a variety of aspects.

2. The specific facts do not involve *ab*.

3. The circumscription of *ab* is done with all predicates variable. This means that the axioms must be sufficient to tie them down.

I had hoped that the simple abnormality formalism would be adequate to express common sense knowledge. Unfortunately, this seems not to be the case. Consider the following axioms.

$$\neg ab\ aspect1\ x \supset \neg flies\ x,$$

$$bird\ x \supset ab\ aspect1\ x,$$

$$bird\ x \wedge \neg ab\ aspect2\ x \supset flies\ x,$$

$$canary\ x \wedge \neg ab\ aspect3\ x \supset bird\ x,$$

$$canary\ Tweety.$$

We ask whether Tweety flies. Simply circumscribing *ab* leaves this undecided, because Tweety can either be abnormal in *aspect*1 or in *aspect*3. Common sense tells us that we should conclude that Tweety flies. This can be achieved by preferring to have Tweety abnormal in *aspect*1 to having Tweety abnormal in *aspect*3. It is not yet clear whether this can be done using the *simple circumscriptive formalism.* Our approach to solving this problem is discussed in the following section on prioritized circumscription. However, simple abnormality theories may be adequate for an interesting set of common sense axiomatizations.

12 PRIORITIZED CIRCUMSCRIPTION

An alternate way of introducing formula circumscription is by means of an ordering on tuples of predicates satisfying an axiom. We define $P \leq P'$ by

$$\forall PP'.P \leq P' \equiv \forall x.E(P,x) \supset E(P',x). \tag{46}$$

That $P0$ is a relative minimum in this ordering is expressed by

$$\forall P.P \leq P0 \supset P = P0, \tag{47}$$

where equality is interpreted extensionally, i.e. we have

$$\forall PP'.P = P' \equiv (\forall x.E(P,x) \equiv E(P',x)). \tag{48}$$

Assuming that we look for a minimum among predicates P satisfying $A(P)$, (46) expands to precisely to the circumscription formula (1). In some earlier expositions of circumscription this ordering approach was used, and Vladimir Lifschitz in a recent seminar advocated returning to it as a more fundamental and understandable concept.

I'm beginning to think he's right about it being more understandable, and there seems to be a more fundamental reason for using it. Namely, certain common sense axiomatizations are easier to formalize if we use a new kind of ordering, and circumscription based on this kind of ordering doesn't seem to reduce to ordinary formula circumscription.

We call it *prioritized circumscription.*

Suppose we write some bird axioms in the form

$$\forall x.\neg ab\ aspect1\ x \supset \neg flies\ x \tag{49}$$

and

$$\forall x.bird\ x \wedge \neg ab\ aspect2\ x \supset ab\ aspect1\ x. \tag{50}$$

The intent is clear. The goal is that being a bird and not abnormal in *aspect*2 prevents the application of (49). However, circumscribing $ab\ z$ with the conjunction of (49) and (50) as $A(ab)$ doesn't have this effect, because (50) is equivalent to

$$\forall x.bird\ x \supset ab\ aspect1\ x \vee ab\ aspect2\ x, \tag{51}$$

and there is no indication that one would prefer to have $aspect1\ x$ abnormal rather than to have $aspect2\ x$ abnormal. Circumscription then results in a disjunction which is not wanted in this case. The need to avoid this disjunction is why the axioms in section 5 (page 6) included cancellation of inheritance axioms.

However, by using a new kind of ordering we can leave (49) and (50) as is, and still get the desired effect.

We define two orderings on *ab* predicates, namely

$$\forall ab\ ab'.ab\ \le_1 ab' \equiv \forall x.ab\ aspect1\ x \supset ab'\ aspect1\ x \tag{52}$$

and

$$\forall ab\ ab'.ab\ \le_2 ab' \equiv \forall x.ab\ aspect2\ x \supset ab'\ aspect2\ x. \tag{53}$$

We then combine these orderings lexicographically giving $\le_2$ priority over $\le_1$ getting

$$\forall ab\ ab'.ab\ \le_{1<2} ab' \equiv ab\ \le_2 ab' \wedge ab\ =_2 ab' \supset ab\ \le_1 ab'. \tag{54}$$

Choosing *ab*0 so as to minimize this ordering yields the result that exactly birds can fly. However, if we add

$$\forall x.ostrich\ x \supset ab\ aspect2\ x, \tag{55}$$

we'll get that ostriches (whether or not ostriches are birds) don't fly without further axioms. If we use

$$\forall x.ostrich\ x \wedge \neg ab\ aspect3\ x \supset ab\ aspect2\ x \tag{56}$$

instead of (55), we'll have to revise our notion of ordering to put minimizing *ab aspect*3 *x* at higher priority than minimizing *aspect*2 *x* and *a fortiori* at higher priority than minimizing *aspect*1.

This suggests providing a partial ordering on aspects giving their priorities and providing axioms that permit deducing the ordering on *ab* from the sentences that describe the ordering relations. Lifschitz (1985) further develops the idea of prioritized circumscription.

I expect that *prioritized circumscription* will turn out to be the most natural and powerful variant.

Simple abnormality theories seem to be inadequate also for the blocks world described in section 11. I am indebted to Lifschitz for the following example. Consider

$$S2 = result(move(B, top\ A), result(move(A, top\ B), S0)), \tag{57}$$

where $S0$ is a situation with exactly blocks A and B on the table. Intuitively, the second action $move(B, top\ A)$ is unsuccessful, because after the first action A is on B, and so B isn't clear. Suppose we provide by a suitable axiom that when the block to be moved is not clear or the destination place is not clear, then the situation is normally unchanged. Then $S2$ should be the same situation as $S1 = result(move(A, B), S0)$. However, simple circumscription of *ab* won't give this result, because the first move is only normally successful, and if the first move is unsuccessful for some unspecified reason, the second move may succeed after all. Therefore, circumscription of *ab* only gives a disjunction.

Clearly the priorities need to be arranged to avoid this kind of unintended "sneak disjunction". The best way to do it by imposing priorities isn't clear at the time of this writing.

13 GENERAL CONSIDERATIONS AND REMARKS

1. Suppose we have a data base of facts axiomatized by a formalism involving the predicate *ab*. In connection with a particular problem, a program takes a

subcollection of these facts together with the specific facts of the problem and then circumscribes $ab\ z$. We get a second order formula, and in general, as the natural number example of (McCarthy 1980) shows, this formula is not equivalent to any first order formula. However, many common sense domains are axiomatizable in such a way that the circumscription is equivalent to a first order formula. In this case we call the circumscription collapsible. For example, Vladimir Lifschitz (1985) has shown that this is true if the axioms are from a certain class he calls "separable" formulas. This can presumably be extended to other cases in which the ranges and domains of the functions are disjoint, so that there is no way of generating an infinity of elements.

Circumscription is also collapsible when the predicates are all monadic and there are no functions.

2. We can then regard the process of deciding what facts to take into account and then circumscribing as a process of compiling from a slightly higher level nonmonotonic language into mathematical logic, especially first order logic. We can also regard natural language as higher level than logic. However, as I shall discuss elsewhere, natural language doesn't have an independent reasoning process, because most natural language inferences involve suppressed premisses which are not represented in natural language in the minds of the people doing the reasoning.

Reiter has pointed out, both informally and implicitly in (Reiter 1982) that circumscription often translates directly into Prolog program once it has been decided what facts to take into account.

3. Circumscription has interesting relations to Reiter's (1980a) logic of defaults. In simple cases they give the same results. However, a computer program using default logic would have to establish the existence of models, perhaps by constructing them, in order to determine that the sentences mentioned in default rules were consistent. Such computations are not just selectively applying the rules of inference of logic but are metamathematical. At present this is treated entirely informally, and I am not aware of any computer program that finds models of sets of sentences or even interacts with a user to find and verify such models.

Circumscription works entirely within logic as Appendices A and B illustrate. It can do this, because it uses second order logic to import some of the model theory of first order formulas into the theory itself. Finding the right substitution for the predicate variables is, in the cases we have examined, the same task as finding models of a first order theory. Putting everything into the logic itself is an advantage as long as there is neither a good theory of how to construct models nor programs that do it.

Notice, however, that finding an interpretation of a language has two parts — finding a domain and interpreting the predicate and function letters by predicates and functions on the domain. It seems that the second is easier to import into second order logic than the first. This may be why our treatment of unique names is awkward.

4. We are only part way to our goal of providing a formalism in which a database of common sense knowledge can be expressed. Besides sets of axioms involving ab, we need ways of specifying what facts shall be taken into account and what functions and predicates are to be taken as variable.

Moreover, some of the circumscriptions have unwanted conclusions, e.g. that there are no ostriches if none are explicitly mentioned. Perhaps some of this can be fixed by introducing the notion of present situation. An axiom that ostriches exist will do no harm if what is allowed to vary includes only ostriches that are present.

5. Nonmonotonic formalisms in general, and circumscription in particular, have many as yet unrealized applications to formalizing common sense knowledge and reasoning. Since we have to think about these matters in a new way, what the applications are and how to realize them isn't immediately obvious. Here are some suggestions.

When we are searching for the "best" object of some kind, we often jump to the conclusion that the best we have found so far is the best. This process can be represented as circumscribing $better(x, candidate)$, where $candidate$ is the best we have found so far. If we attempt this circumscription while including certain information in our axiom $A(better, P)$, where P represents additional predicates being varied, we will succeed in showing that there is nothing better only if this is consistent with the information we take into account. If the attempt to circumscribe fails, we would like our reasoning program to use the failure as an aid to finding a better object. I don't know how hard this would be.

14 APPENDIX A

CIRCUMSCRIPTION IN A PROOF CHECKER

At present there are no reasoning or problem-solving programs using circumscription. A first step towards such a program involves determining what kinds of reasoning are required to use circumscription effectively. As a step towards this we include in this and the following appendix two proofs in EKL (Ketonen and Weening 1984), an interactive theorem prover for the theory of types. The first does the bird problem and the second a simple unique names problem. It will be seen that the proofs make substantial use of EKL's ability to admit arguments in second order logic.

Each EKL step begins with a command given by the user. This is usually followed by the sentence resulting from the step in a group of lines each ending in a semicolon, but this is omitted for definitions when the information is contained in the command. We follow each step by a brief explanation. Of course, the reader may skip this proof if he is sufficiently clear about what steps are involved. However, I found that pushing the proof through EKL clarified my ideas considerably as well as turning up bugs in my axioms.

$$\begin{array}{l}
1.\ (DEFINE\ A\\
\quad |\forall AB\ FLIES.A(AB, FLIES) \equiv\\
\qquad (\forall X.\neg AB(ASPECT1(X)) \supset \neg FLIES(X))\wedge\\
\qquad (\forall X.BIRD(X) \supset AB(ASPECT1(X)))\wedge\\
\qquad (\forall X.BIRD(X) \wedge \neg AB(ASPECT2(X)) \supset FLIES(X))\wedge\\
\qquad (\forall X.OSTRICH(X) \supset AB(ASPECT2(X)))\wedge\\
\qquad (\forall X.OSTRICH(X) \wedge \neg AB(ASPECT3(X)) \supset \neg FLIES(X))|NIL)
\end{array}$$

This defines the second order predicate $A(ab, flies)$, where ab and $flies$ are predicate variables. Included here are the specific facts about flying being taken into account.

$$\begin{array}{l}
;labels : SIMPINFO\\
2.\ (AXIOM\\
\quad |(\forall X\ Y.\neg ASPECT1(X) = ASPECT2(Y))\wedge\\
\quad (\forall X\ Y.\neg ASPECT1(X) = ASPECT3(Y))\wedge\\
\quad (\forall X\ Y.\neg ASPECT2(X) = ASPECT3(Y))\wedge\\
\quad (\forall X\ Y.ASPECT1(X) = ASPECT1(Y) \equiv X = Y)\wedge\\
\quad (\forall X\ Y.ASPECT2(X) = ASPECT2(Y) \equiv X = Y)\wedge\\
\quad (\forall X\ Y.ASPECT3(X) = ASPECT3(Y) \equiv X = Y)|)
\end{array}$$

These facts about the distinctness of aspects are used in step 20 only. Since axiom 2 is labelled SIMPINFO, the EKL simplifier uses it as appropriate when it is asked to simplify a formula.

3. $(DEFINE\ A1$
$|\forall AB\ FLIES.A1(AB, FLIES) \equiv$
$A(AB, FLIES) \wedge$
$(\forall AB1\ FLIES1.A(AB1, FLIES1) \wedge (\forall Z.AB1(Z) \supset AB(Z)) \supset$
$(\forall Z.AB(Z) \equiv AB1(Z)))|NIL)$

This is the circumscription formula itself.

4. $(ASSUME\ |A1(AB, FLIES)|)$
$deps : (4)$

Since EKL cannot be asked (yet) to do a circumscription, we assume the result. Most subsequent statements list line 4 as a dependency. This is appropriate since circumscription is a rule of conjecture rather than a rule of inference.

5. $(DEFINE\ FLIES2|\forall X.FLIES2(X) \equiv BIRD(X) \wedge \neg OSTRICH(X)|NIL)$

This definition and the next say what we are going to substitute for the bound predicate variables.

6. $(DEFINE\ AB2$
$|\forall Z.AB2(Z) \equiv (\exists X.BIRD(X) \wedge Z = ASPECT1(X)) \vee$
$(\exists X.OSTRICH(X) \wedge Z = ASPECT2(X))|NIL)$

The fact that this definition is necessarily somewhat awkward makes for some difficulty throughout the proof.

7. $(RW\ 4\ (OPEN\ A1))$
$A(AB, FLIES) \wedge (\forall AB1\ FLIES1.A(AB1, FLIES1) \wedge$
$(\forall Z.AB1(Z) \supset AB(Z)) \supset (\forall Z.AB(Z) \equiv AB1(Z)))$
$deps : (4)$

This step merely expands out the circumscription formula. RW stands for "rewrite a line", in this case line 4.

8.$(TRW\ |A(AB, FLIES)|(USE\ 7))$
$A(AB, FLIES)$
$deps : (4)$

We separate the two conjuncts of 7 in this and the next step.

9. $(TRW\ |\forall AB1\ FLIES1.A(AB1, FLIES1)\wedge$
$(\forall Z.AB1(Z) \supset AB(Z)) \supset (\forall Z.AB(Z) \equiv AB1(Z))|$
$(USE\ 7))$
$\forall AB1\ FLIES1.A(AB1, FLIES1) \wedge (\forall Z.AB1(Z) \supset AB(Z)) \supset$
$(\forall Z.AB(Z) \equiv AB1(Z))$
$deps$: (4)

10. $(RW\ 8\ (OPEN\ A))$
$(\forall X.\neg AB(ASPECT1(X)) \supset \neg FLIES(X))\wedge$
$(\forall X.BIRD(X) \supset AB(ASPECT1(X)))\wedge$
$(\forall X.BIRD(X) \wedge \neg AB(ASPECT2(X)) \supset FLIES(X))\wedge$
$(\forall X.OSTRICH(X) \supset AB(ASPECT2(X)))\wedge$
$(\forall X.OSTRICH(X) \wedge \neg AB(ASPECT3(X)) \supset \neg FLIES(X))$
$deps$: (4)

Expanding out the axiom using the definition a in step 1.

11. $(ASSUME\ |AB2(Z)|)$
$deps$: (11)

Our goal is step 15, but we need to assume its premiss and then derive its conclusion.

12. $(RW11\ (OPEN\ AB2))$
$(\exists X.BIRD(X) \wedge Z = ASPECT1(X))\vee$
$(\exists X.OSTRICH(X) \wedge Z = ASPECT2(X))$
$deps$: (11)

We use the definition of ab.

13. $(DERIVE\ |AB(Z)|\ (12\ 10)\ NIL)$
$AB(Z)$
$deps$: (4 11)

This is our first use of EKL's DERIVE command. It is based on the notion of direct proof of (Ketonen and Weyhrauch 1984). Sometimes it can do rather complicated things in one step.

14. $(CI\ (11)\ 13\ NIL)$
$AB2(Z) \supset AB(Z)$
$deps : (4)$

We discharge the assumption 11 with the "conditional introduction" command.

15. $(DERIVE\ |\forall Z.AB2(Z) \supset AB(Z)|\ (14)\ NIL)$
$\forall Z.AB2(Z) \supset AB(Z)$
$deps : (4)$

Universal generalization.

16. $(DERIVE$
$|(\forall X.\neg AB2(ASPECT1(X)) \supset \neg FLIES2(X))\wedge$
$(\forall X.BIRD(X) \supset AB2(ASPECT1(X)))\wedge$
$(\forall X.BIRD(X) \wedge \neg AB2(ASPECT2(X)) \supset FLIES2(X))\wedge$
$(\forall X.OSTRICH(X) \supset AB2(ASPECT2(X)))\wedge$
$(\forall X.OSTRICH(X) \wedge \neg AB2(ASPECT3(X)) \supset$
$\neg FLIES2(X))|()(OPEN\ AB2\ FLIES2))$
$;(\forall X.\neg AB2(ASPECT1(X)) \supset \neg FLIES2(X))\wedge$
$;(\forall X.BIRD(X) \supset AB2(ASPECT1(X)))\wedge$
$;(\forall X.BIRD(X) \wedge \neg AB2(ASPECT2(X)) \supset FLIES2(X))\wedge$
$;(\forall X.OSTRICH(X) \supset AB2(ASPECT2(X)))\wedge$
$;(\forall X.OSTRICH(X) \wedge \neg AB2(ASPECT3(X)) \supset \neg FLIES2(X))$

This is another rather lengthy computation, but it tells us that $ab2$ and $flies2$ satisfy the axioms for ab and flies.

17. $(UE\ ((AB.|AB2|)\ (FLIES.|FLIES2|))\ 1\ NIL)$
$;A(AB2, FLIES2) \equiv$
$;(\forall X.\neg AB2(ASPECT1(X)) \supset \neg FLIES2(X))\wedge$
$;(\forall X.BIRD(X) \supset AB2(ASPECT1(X)))\wedge$
$;(\forall X.BIRD(X) \wedge \neg AB2(ASPECT2(X)) \supset FLIES2(X))\wedge$
$;(\forall X.OSTRICH(X) \supset AB2(ASPECT2(X)))\wedge$
$;(\forall X.OSTRICH(X) \wedge \neg AB2(ASPECT3(X)) \supset \neg FLIES2(X))$

Now we substitute $ab2$ and $flies2$ in the definition of A and get a result we can compare with step 16.

18. $(RW\ 17\ (USE\ 16))$
$; A(AB2, FLIES2)$

We have shown that $ab2$ and $flies2$ satisfy A.

19. $(DERIVE\ |\forall Z.AB(Z) \equiv AB2(Z)|\ (9\ 15\ 18)\ NIL)$
$; \forall Z.AB(Z) \equiv AB2(Z)$
$; deps : (4)$

9 was the circumscription formula, and 15 and 18 are its two premisses, so we can now derive its conclusion. Now we know exactly what entities are abnormal.

20. $(RW\ 8\ ((USE\ 1\ MODE:\ EXACT)$
$((USE\ 19\ MODE:\ EXACT)\ (OPEN\ AB2))))$
$; (\forall X.\neg(\exists X1.BIRD(X1) \wedge X = X1) \supset \neg FLIES(X)) \wedge$
$; (\forall X.BIRD(X) \wedge \neg(\exists X2.OSTRICH(X2) \wedge X = X2) \supset FLIES(X)) \wedge$
$; (\forall X.OSTRICH(X) \supset \neg FLIES(X))$
$; deps : (4)$

We rewrite the axiom now that we know what's abnormal. This gives a somewhat awkward formula that nevertheless contains the desired conclusion. The occurrences of equality are left over from the elimination of the aspects that used the axiom of step 2.

21. $(DERIVE\ |\forall X.FLIES(X) \equiv$
$BIRD(X) \wedge \neg OSTRICH(X)|\ (20)\ NIL)$
$; \forall X.FLIES(X) \equiv BIRD(X) \wedge \neg OSTRICH(X)$
$; deps : (4)$

DERIVE straightens out 20 to put the conclusion in the desired form. The result is still dependent on the assumption of the correctness of the circumscription made in step 4.

Clearly if circumscription is to become a practical technique, the reasoning has to become much more automatic.

15 APPENDIX B

Here is an annotated EKL proof that circumscribes the predicate $e(x,y)$ discussed in section 6.

$$(proof\ unique\ names)$$

What the user types is indicated by the numbered statements in lower case. What EKL types is preceded by semicolons at the beginning of each line and is in upper case. We omit EKL's type-out when it merely repeats what the command asked it to do, as in the commands DERIVE, ASSUME and DEFINE.

1. $(axiom\ |index\ a = 1 \wedge index\ b = 2 \wedge index\ c = 3 \wedge index\ d = 4|)$

Since EKL does not have attachments to determine the equivalence of names, we establish a correspondence between the names in our domain and some natural numbers.

2. $(derive|\neg(1=2) \wedge \neg(1=3) \wedge \neg(2=3) \wedge \neg(1=4) \wedge \neg(2=4) \wedge$
$\neg(3=4)|)$

EKL does know about the distinctness of natural numbers, so this can be derived.

$$(der - slow)$$

We have to tell EKL to use the properties of equality rather than regarding it as just another predicate symbol in order to do the next step. Sometimes this leads to combinatorial explosion.

3. $(derive\ |a \neq\ b|\ (1\ 2))$

This shows that two names themselves are distinct.

4. $(define\ equiv\ |\forall e.equiv\ e \equiv (\forall x.e(x,x)) \wedge (\forall x\ y.e(x,y) \supset e(y,x))$
$\wedge(\forall x\ y\ z.e(x,y) \wedge e(y,z) \supset e(x,z))|)$

Here we use second order logic to define the notion of equivalence relation. The first word after "define" is the entity being defined and included between vertical bars is the defining relation. EKL checks that an entity satisfying the relation exists.

$$5.\quad (define\ ax\ |\forall e.ax\ e \equiv e(a,b) \wedge equiv\ e|)$$

We define ax as a predicate we want our imitation equality to satisfy. We have chosen a very simple case, namely making a and b "equal" and nothing else.

$$6.\quad (define\ ax1|\forall e.ax1\ e \equiv ax\ e \wedge \forall e1.(ax\ e1 \wedge (\forall\ x\ y.e1(x,y) \supset e(x,y))$$
$$\supset (\forall x\ y.e(x,y) \equiv e1(x,y)))|)$$

This defines $ax1$ as the second order predicate specifying the circumscription of ax.

$7.\quad (assume\ |ax1(e0)|)$
$(label\ circum)$

We now specify that $e0$ satisfies $ax1$. It takes till step 17 to determine what $e0$ actually is. When EKL includes circumscription as an operator, we may be able to write something like circumscribe$(e0, ax1)$ and make this step occur. For now it's just an ordinary assumption.

$$8.\quad (define\ e2\ |\forall x\ y.e2(x,y) \equiv (x = a \wedge y = b) \vee (x = b \wedge y = a)$$
$$\vee x = y|)$$

The predicate $e2$ defined here is what $e0$ will turn out to be.

$$9.\quad (derive\ |equive2|\ nil\ (open\ equiv)\ (open\ e2))$$

Now EKL agrees that $e2$ is an equivalence relation. This step takes the KL-10 about 14 seconds.

$10.\quad (derive\ |ax\ e2|\ (9)\ (open\ ax)\ (open\ e2))$
$(label\ ax\ e2)$

Moreover it satisfies ax.

$11.\quad (rw\ circum\ (open\ ax1))$
$;AX(E0) \wedge (\forall\ E1.AX(E1) \wedge (\forall X\ Y.E1(X,Y) \supset E0(X,Y)) \supset$
$;(\forall X\ Y.E0(X,Y) \equiv E1(X,Y)))$
$;deps : (CIRCUM)$

A trivial step of expanding the definition of $ax1$. EKL tells us that this fact depends on the assumption CIRCUM. So do many of the subsequent lines of the proof, but we omit it henceforth to save space.

$$\begin{array}{l} 12. \quad (trw\ |ax\ e0|\ (use\ 11)) \\ ;AX(E0) \end{array}$$

The first conjunct of the previous step.

$$\begin{array}{l} 13. \quad (rw\ 12\ (open\ ax\ equiv)) \\ (label\ fact1) \\ ;E0(A,B) \land (\forall X.E0(X,X)) \land (\forall X\ Y.E0(X,Y) \supset E0(Y,X)) \land \\ ;(\forall X\ Y\ Z.E0(X,Y) \land E0(Y,Z) \supset E0(X,Z)) \end{array}$$

We expand $ax(e0)$ according to the definitions of ax and $equiv$.

$$\begin{array}{l} 14. \quad (derive\ |\forall p\ q\ r.(p \lor q \supset r) \equiv (p \supset r) \land (q \supset r)|) \\ (label\ rewrite\ by\ cases) \end{array}$$

This is a fact of propositional calculus used as a rewrite rule in the next step. A program that can use circumscription by itself will either need to generate this step or systematically avoid the need for it.

$$\begin{array}{l} 15. \quad (trw\ |e2(x,y) \supset e0(x,y)| \\ \qquad ((open\ e2)\ (use\ rewrite\ by\ cases\ mode:\ always)\ (use\ fact1))) \\ ;E2(X,Y) \supset E0(X,Y) \end{array}$$

This is the least obvious step, because rewrite by cases is used after some preliminary transformation of the formula.

$$16. \quad (derive\ |\forall x\ y.e0(x,y) \equiv e2(x,y)|\ (ax\ e2\ 11\ 15))$$

DERIVE is substituting $e2$ for the variable $e1$ in step 11 and using the fact $ax(e2)$ and step 15 to infer the conclusion of the implication that follows the quantifier $\forall e$.

$$\begin{array}{l} 17. \quad (rw\ 16\ (open\ E2)) \\ ;\forall X\ Y.E0(X,Y) \equiv X = A \land Y = B \lor X = B \land Y = A \lor X = Y \\ ;deps:\ (CIRCUM) \end{array}$$

Expanding the definition of $e2$ tells us the final result of circumscribing $e0(x,y)$. A more complex $ax(e0)$ — see step 5 — would give a more complex result upon circumscription. However, it seems that the proof would be similar. Therefore, it could perhaps be made into some kind of macro.

16 Acknowledgments

I have had useful discussions with Matthew Ginsberg, Benjamin Grosof, Vladimir Lifschitz and Leslie Pack[1]. The work was partially supported by NSF and by DARPA. I also thank Jussi Ketonen for developing EKL and helping me with its use. In particular he greatly shortened the unique names proof.

17 References

Etherington, D., Mercer, R. and Reiter, R. (1985). On the Adequacy of Predicate Circumscription for Closed-World Reasoning, *Computational Intelligence 1.*

Ketonen, Jussi and Joseph S. Weening (1984). *EKL — An Interactive Proof Checker, User's Reference Manual*, Computer Science Department, Stanford University.

Ketonen, Jussi and Richard W. Weyhrauch (1984). A Decidable Fragment of Predicate Calculus, accepted for publication in the *Journal for Theoretical Computer Science.*

Lifschitz, Vladimir (1985). Computing Circumscription in *Proc. IJCAI-85.*

McCarthy, John (1959). Programs with Common Sense, in *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, London: Her Majesty's Stationery Office.

McCarthy, John and Patrick Hayes (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence, in B. Meltzer and D. Michie (eds), *Machine Intelligence 4*, Edinburgh University. (Reprinted in B. L. Webber and N. J. Nilsson (eds.), *Readings in Artificial Intelligence*, Tioga, 1981, pp. 431–450; also in M. J. Ginsberg (ed.), *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann, 1987, pp. 26–45)

McCarthy, John (1977). Epistemological Problems of Artificial Intelligence, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, M.I.T., Cambridge, Mass. (Reprinted in B. L. Webber and N. J.

[1] Leslie P. Kaelbling

Nilsson (eds.), *Readings in Artificial Intelligence*, Tioga, 1981, pp. 459–465; also in M. J. Ginsberg (ed.), *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann, 1987, pp. 46–52)

McCarthy, John (1979). First Order Theories of Individual Concepts and Propositions, in Michie, Donald (ed.) *Machine Intelligence 9*, Ellis Horwood.

McCarthy, John (1980). Circumscription — A Form of Non-Monotonic Reasoning, *Artificial Intelligence*, Volume 13, Numbers 1,2. (Reprinted in B. L. Webber and N. J. Nilsson (eds.), *Readings in Artificial Intelligence*, Tioga, 1981, pp. 466–472; also in M. J. Ginsberg (ed.), *Readings in Nonmonotonic Reasoning*, Morgan Kaufmann, 1987, pp. 145–152)

McCarthy, John (1982). Common Business Communication Language, in *Textverarbeitung und Bürosysteme*, Albert Endres and Jürgen Reetz, eds. R. Oldenbourg Verlag, Munich and Vienna 1982.

Reiter, Raymond (1980a). A Logic for Default Reasoning, *Artificial Intelligence*, Volume 13, Numbers 1,2, April.

Reiter, Raymond (1980b). Equality and domain closure in first order data bases, *J. ACM*, **27**, Apr. 1980, 235-249.

Reiter, Raymond (1982). Circumscription Implies Predicate Completion (Sometimes), *Proceedings of the National Conference on Artificial Intelligence, AAAI-82*, William Kaufman, Inc.