CIRCUMSCRIPTION

Circumscription (due to John McCarthy) is one type of nonmonotonic-reasoning formalism. Here is a very general formulation given by Vladimir Lifschitz, where K(A,**Z**) is a given knowledge base (set of wffs) involving predicate letters including A and those in the tuple **Z**, and p and **z** designate predicate variables (in a second-order logic – don't let this abstraction scare you, it will all make sense when we come to an example):

CIRC(K,A,**Z**): K(A,**Z**) & -Exist a **z** [K(a,**z**) & a<A]

Intuition: CIRC extends the KB with an extra conjunct telling us that A is "minimal", in the sense that no substitutes a and **z** for A and **Z** can satisfy K such that a applies only to a proper subset of what A applies to (that's the a<A part). That is, A already applies to as small a set as possible.

We can rewrite the above conjunct as follows, which may make this clearer:

Forall a **z** [{ K(a,**z**) & Forall x (a(x) -> A(x)) } → Forall x (A(x) <--> a(x))]

Thus if we can find specific wffs – let's call them A' and **Z**' – to take the place of the variables a and **z** (and thus to represent alternatives to A and **Z**), that satisfy K(A',**Z**') and such that A' is "contained" in A, then according to CIRC, A' and A are actually equivalent (A is already as "shrunk-down" as A').

That is, we pick (guess at) particular choices A' and Z' for the variables a and **z** and see what results. If we pick right, then by CIRC we can conclude A is small, like A'. But why should A be small (apply to only a few things)?

In many applications, A is used to stand for an "abnormal" property, one that typical items (of a certain class) do not have; then one expects A to apply to very few things, and CIRC is a way to formalize this to force A to be as small as it can be, given the information in K.

Restated: there are three steps involved in applying CIRC: *imagine, reduce, use:*

(i) *imagine* workable precisifications of some key predicate symbols
(ii) show A's precisification is *reductive* – it shrinks what A can mean
(iii) after that, according to CIRC, we may *use* A' <-> A.

In simple cases, it can work like this example: let K(Abnormal, Flies, Penguin) consist of background knowledge such as

1. (Forall x) [(Bird(x) & -Abnormal(x)) → Flies(x)]
2. (Forall x) [Penguin(x) → -Flies(x)]

3. (Forall x) [Penguin(x) → Bird(x)]

and specific knowledge such as

4. Bird(tweety)
5. Penguin(penny)
6. tweety ≠ penny

Note the use of the Forall quantifier; despite that, the method (still to be described) manages to avoid the conclusion that all birds fly, because of the predicate Abnormal. This prevents applying wff 1 above to the case of x=tweety, unless we also have the knowledge that -Abnormal(tweety). Here is how circumscription achieves that (starting with an intuitive description first, and then something precise):

We imagine that we do not know in full detail the exact meaning of commonsense predicates (such as Bird, Penguin, Abnormal, Flies); in fact, we do not, as indicated in the lecture notes (so-called natural kinds -- categories of things occurring in the real world -- tend to be ill-defined and/or vague). So, we can try out various possible "precisifications" of Bird, imagining it to mean this, or that, until we find a meaning that "works for us". In particular, this form of circumscription focuses on the predicate Abnormal. If we knew which birds were abnormal, we'd then know whether Tweety is abnormal, and then we'd know whether we can apply wff 1 when x=tweety. In this example no non-birds are mentioned at all, so the meaning of Bird is not an issue; but Flies and Penguin and Abnormal are all “up for grabs” so to speak. Abnormal is the one of interest, since the idea is that most things are not abnormal, so we look for a precisification of it that applies to as few things as possible. Of course, restricting Abnormal will require restricting Penguin as well (since by 2 and 3 penguins are non-flying birds and by 1 this forces them to be abnormal.) And restricting Abnormal enlarges –Abnormal, and so (by 1) will enlarge Flies. So we need to consider various possible precisifications of Flies and Penguin as well as of Abnormal. (Thus in this example, **Z** consists of the two predicate symbols Flies and Penguin.)

What might that mean? Well, if we try out a meaning, say some new wff Ab’, that might capture a good meaning for Abnormal (or Ab for short), we'd want this substitution still to satisfy all the things we believe about birds, at least for some possible precisifications of **Z** (i.e., of Flies and Penguin). That is, if we take KB(Ab, Flies, Penguin) to stand for the entire KB, then we'd want to know that KB(Ab’, Flies’, Penguin’) -- the substitution of precisified versions into KB -- is true. (We’d of course need to choose suitable wffs for Flies’ and Penguin’ too; we’ll call them F’ and P’ for short.) For if not, we'd be considering Ab to have a meaning that violates something we believe about it.

In other words, we'd want to know – based on 1-6, i.e., KB(Ab, F, P) – that

7. KB(Ab',F', P')

But there might be many possible wffs Ab' like that. One of course is just Ab itself. How could we decide on a useful choice of Ab'? Well, here again is the central idea: while there is no magic way to pick out a good meaning for predicates in general (since the world is so complicated), nevertheless when the situation is one of trying to express a default rule -- that is, something that is usually safe to assume true -- we expect the exceptions (the abnormal cases) to be quite limited.

So: we try to choose a substitute Ab' for Ab that seems to be as restrictive as we can make it. In particular, it had better be at least as restrictive as Ab itself:

8. (Forallx) [Ab'(x) → Ab(x)]

If we can find such an Ab', then we use it for Ab itself. Put differently, we now assert that Ab(x) iff Ab'(x), i.e., our Ab' is the "real" (or default) meaning of Abnormal.

Here is a simpler example of a knowledge base K (not the one above, which is further described later below and will be for homework):

A. (Forall x) [(Bird(x) & -Abnormal(x)) → Flies(x)]
B. Abnornal(x) -> -Flies(x)
C. Bird(tweety)
D. Abnormal(penny)
E. tweety ≠ penny

We are not even told here that penny is a bird; so we might want to consider letting the meaning of Bird shift about, as well as those of Ab and F. We want Ab to be small, but it has to apply to penny, so it cannot be empty. So we try Ab'x <-> x=penny, i.e., penny might be the only abnormal entity; and certainly penny won't fly (by axiom B.) This means F has to be restricted, and one easy choice is this: F'(x) <-> x=/=penny. As for Bird, let's try leaving it alone (B' will just be B itself) and see what happens.

To apply CIRC, we need to check that K(A',**Z**') – to show that our choices do not conflict with our axioms – and then that A' shrinks A. After that, CIRC tells us that A really is shrunken like A' after all.

Replacing Ab and F by Ab' (x=penny) and F' (x=/=penny) makes items A., B., and D. above trivially true, and items C. and E. do not change. So this handles part (i) of the three parts needed. Part (ii) is reduction; but since D. tells us Ab(penny) then clearly Ab' -> Ab.

Finally, we now are allowed to use part (iii) to se what results. Taking Ab to be x=penny in A. above, we get
Bird(x) & x=/=penny → Flies(x)

But since we have tweety=/=penny and also Bird(tweety), this yields Flies(tweety), the conclusion we wanted.

We now consider briefly an even simpler example (using B and F as abbreviations for Bird and Flies):

B(x) & -Ab(x) -> F(x)
B(tweety)

We'd like to be able to show F(tweety). More generally, we'd hope to be able – in this particular case – to show that all birds fly, since no birds are known to be abnormal. We can do this by shrinking Ab to be empty/false: Ab'x <-.> x=/=x. But then to verify the *imagine* phase, we need to verify this Ab' works in our two axioms. Alas, we can't show Bx & x=x -> F'x, unless we pick F'x simply as Bx. But we can do this! And since Ab'x -> Ab(x) (vacuously, the LHS is always false), then CIRC applies and we get Ab(x) <-> x=/=x, so –Ab(x) is always true, and the first axiom reduces to Bx -> Fx: all birds fly. In particular, Flies(tweety) results.

There remain some unsatisfactory aspects to this approach. For instance, we'd like an automated agent to be able to pick which predicate symbols **Z** to allow to vary (right now a human has to do this). And there are other more open issues as well, some having to do with getting conclusions that are far too strong (hinted at in what we just saw: all birds fly, even though the very mention of abnormal birds should be a strong hint that this is not true).

Now you try out the more complex case earlier, with axioms 1-6 – this is your next *homework set* – (and I will GIVE you the minimal Ab' to use, as well as good substitutions for Penguin and Flies; but YOU have to show it works): Let Ab'(x) be the very simple wff x=penny, let P'(x) be x=penny, and F'(x) be x ≠ penny. This means we are imagining both Abnormal and Penguin to apply to only one thing: the one bird penny; and Flies to apply to everything else. Then do (i), (ii), and (iii) below. (You may give informal proofs in mathematical English, and of course using the original KB items 1-6 as axioms.)

(i) Using 1-6, prove 7. (This shows that x=penny is at least a possible interpretation for Abnormal(x), that is consistent with our KB).

(ii) Prove 8. (This shows taking Ab'(x) to be x=penny is at least as "shrunken" as Abnormal(x) itself; and then from CIRC we can use Ab' in place of Ab; see below.)

(iii) Using 1-6 and Ab' <--> Ab (that is, taking Abnormal to be the same as x=penny) prove the following (this shows that we get the intuitive sort of default conclusions we want):
9. Bird(penny)
10. -Flies(penny)

11. -Abnormal(tweety)
12. Flies(tweety)
13.-Penguin(tweety)