Today

- **Closed World Assumption**
- **Reasoning with defaults**

See Nilsson and Genesereth, chapter 6; also in Russell and Norvig, ch 10.

Knowing the answers

A good situation to be in is where we have enough information to answer any possible query. If we know

$$poor(jane)$$
$$poor(jane) \rightarrow happy(jane)$$
$$happy(fred)$$

we do not know enough to answer the query

$$?-\ poor(fred)$$

Complete Theories

We say a theory/KB is *complete* (for ground atoms) iff for every query (like $poor(fred)$) we can conclude either $poor(fred)$ or $\neg poor(fred)$.

A ground atom is a statement of the form $P(t_1, \ldots, t_n)$ where there are no variables in any t_i; so it is a basic statement about particular objects.

Our example KB is not complete in this sense; we can extend it to make a complete KB using the Closed World Assumption (CWA). The idea is to add in the *negation* of a ground atom whenever the ground atom cannot be deduced from the KB.

This makes the assumption that

> *all the basic positive information about the domain follows from what is already in the KB.*

CWA as an augmented KB

We can define the effect of the CWA using the standard logic we saw earlier. Given a KB written in first-order logic, we augment KB to get a bigger set of formulas $CWA(KB)$; the extra formulas we add are:

$$X_{KB} = \{\ \neg p(t_1, \ldots, t_n) : \textbf{not}\ KB \vdash p(t_1, \ldots, t_n)\ \}$$

Now we can define what it is to follow from a KB using CWA: a formula Q follows from KB using the CWA iff

$$KB \cup X_{KB} \models Q$$

Example

In the example, we can now conclude $\neg poor(fred)$, since from the original KB we *cannot* show $poor(fred)$. Thus we have $\neg poor(fred)$ is in X_{KB}.

In fact, in this case

$$X_{KB} = \{ \neg poor(fred) \},$$

assuming there are no other constants in the language except $jane, fred$. In this case, we can compute the set X_{KB} by looking at all possibilities.

One use of CWA is in looking at a failed Prolog query of the form

?- property(t1,t2).

as saying that the query is in fact false.

CWA and databases

It is standard to use the CWA in databases. Suppose we have a list of neighbouring Scottish councils:

```
nextTo( Clackmannan, Falkirk ).
nextTo( Clackmannan, Stirling ).
nextTo( Falkirk, West Lothian ).
⋮
```

If all the information is positive, we need the CWA to conclude that East Lothian and Falkirk are not neighbours.

There is a choice of vocabulary here – we could have chosen a database of non-neighbours, and used CWA the opposite way round. In fact, normally the more informative term is used for the positive concept, as here.

CWA and Horn clauses

A definite clause is a formula of the shape

$$P_1 \wedge \cdots \wedge P_n \rightarrow Q$$

where the P_i and Q are atomic statements, maybe with variables;
there may be any number (even none) of P_i, and Q is always there.

One reason why CWA is often used with KB expressed in a Prolog-like way is the following result. If KB consists of definite clauses, then the augmented KB $CWA[KB]$ is consistent; that is, there is some interpretation of the language under which all the formulas in $CWA[KB]$ are true.

Logic and monotonic reasoning

It's a basic feature of standard logic that it is *monotonic*: if we add new assumptions to a theory, we never invalidate any conclusions we could already make.

In other words, if Q follows logically from a set of statements KB, and X is a set of statements, then Q follows from KB together with X.

If $KB \models Q$, **then** $KB \cup X \models Q$

Reasoning with the CWA does not have this property; we say it is *non-monotonic*. Adding extra information can invalidate earlier conclusions.

Example

From our toy example, form a new KB by adding $poor(fred)$ to get the new KB':

$$\begin{array}{rcl} & & poor(jane) \\ poor(jane) & \rightarrow & happy(jane) \\ & & happy(fred) \\ & & poor(fred) \end{array}$$

Now $\neg poor(fred)$ is not in $X_{KB'}$, and so we do not have $CWA[KB] \models \neg poor(fred)$ any more.

CWA may be inconsistent!

Beware that CWA of KB may be inconsistent, even when KB is consistent. For example, take the KB to have a single statement $british(louise) \vee french(louise)$, and look at the augmented KB:

we cannot show $british(louise)$, so $\neg british(louise)$ is in X_{KB}.
we cannot show $french(louise)$, so $\neg french(louise)$ is in X_{KB}.

So $CWA[KB]$ has three statements

$$\{\ british(louise) \vee french(louise),\ \neg british(louise),\ \neg french(louise)\ \}$$

and it is impossible for all three to be true.

Note that the initial KB is *not* made of definite clauses.

CWA: use with care

CWA is a strong assumption to make.

It should only be used where it is reasonable to think that all basic positive information is derivable: $british(louise) \vee french(louise)$ is not good enough, because one of the possibilities is true, but *not* derivable.

It should not be used wherever it introduces inconsistency.

Default reasoning

Some everyday reasoning uses default inference – some conclusions are reached *by default* when we do not have full information available. For example:

> Tweety is a bird.
> Typically, birds can fly.
> In the absence of other information, we conclude that Tweety can fly.

Note that this is *not* an argument using probability (though probabilistic arguments are also possible).

Recall: taxonomic hierarchy

We can take a very small subset of FOL and use it to represent hierarchies. Just use

- *predicates* with one argument (for the classes of the hierarchy)
- all statements are either
  - *atomic*: for objects of the class, *or*
  - of the form $A(x) \rightarrow B(x)$, saying that one class is a sub-class of another.

There is implicitly a universal quantifier in the sub-class rule.

Example

Tweety is a bird.
All birds are things.
Ostriches are birds.
Flying ostriches are ostriches.

Represent as:

$$\begin{array}{rcl} & & bird(tweety) \\ bird(X) & \rightarrow & thing(X) \\ ostrich(X) & \rightarrow & bird(X) \\ flying\_ostrich(X) & \rightarrow & ostrich(X). \end{array}$$

What can we conclude?

Using standard logic, what can we conclude from the statements we have?

We can conclude $thing(tweety)$; "thing" is the most general class, and usually the hierarchy is arranged so that every entity is a thing.

We *cannot* conclude $flies(tweety)$, or $\neg flies(tweety)$; we should not add $bird(X) \rightarrow flies(X)$ to the hierarchy, because some birds do not fly. Default logic was proposed by Ray Reiter to deal with this situation.

Default Rules

Default rules are used to deal with the situation where we want to say that something C follows from something else A by default, provided that as far as we know something else again B may be true.

We write

$$\frac{A(X) : B(X)}{C(X)}$$

to say that if A holds (for some object(s)), and we do not know that B is false, then we can conclude that C holds.

More precisely, if there is a term (without variables) such that $A(t)$ can be derived and $B(t)$ *cannot* be derived, then we conclude $C(t)$. So in our example, use the rule

$$\frac{bird(X) : flies(X)}{flies(X)}$$

Using default rules

We now have:

- $bird(tweety)$ by assumption
- we cannot derive $\neg flies(tweety)$, as we saw before
- therefore we conclude $flies(tweety)$.

This sort of rule is called a *normal default rule*, since it has the shape

$$\frac{A(X) : C(X)}{C(X)}$$

This is the most common use of default rules.

Logic and monotonic reasoning

It's a basic feature of standard logic that it is *monotonic*: if we add new assumptions to a theory, we never invalidate any conclusions we could already make.

In other words, if Q follows logically from a set of statements KB, and X is a set of statements, then Q follows from KB together with X.

$$\textbf{If } KB \models Q\textbf{, then } KB \cup X \models Q$$

Default reasoning does not have this property; we say it is *non-monotonic*. Adding extra information can invalidate earlier conclusions.

Non-monotonic example

Let's take a starting KB:

$$\begin{array}{rcl} & & bird(tweety) \\ bird(X) & \rightarrow & thing(X) \\ ostrich(X) & \rightarrow & bird(X) \\ flying\_ostrich(X) & \rightarrow & ostrich(X) \\ ostrich(X) & \rightarrow & \neg flies(X) \end{array}$$

and the default rule

$$\frac{bird(X) : flies(X)}{flies(X)}$$

We can still deduce that $flies(tweety)$ (check this)

Example continued

Now suppose we add to the KB (we find out that) Tweety is an ostrich:

$$ostrich(tweety)$$

Now we can no longer conclude that $flies(tweety)$ – that's just as well, otherwise the KB would have become inconsistent.

We can no longer use the default rule

$$\frac{bird(X) : flies(X)}{flies(X)}$$

since now we can show $\neg flies(tweety)$.

So, default logic is a non-monotonic logic.

Summary

- Non-monotonic inference
- Closed World Assumption
- Default rules