Recursive rules

Let us add another relation to our family program, the `ancestor` relation.

If we consider again the family tree page 16, we model the `ancestor` binary relation by saying that `X` is an ancestor of `Y` if there is a chain of parenthood relationships from `X` to `Y`. For example, Pam is an ancestor of Jim and Tom is an ancestor of Liz.

The latter example actually illustrates a special case of the `ancestor` relation, when `X` is a parent of `Y`. This case is very easy to specify in Prolog:

```
ancestor(X,Y) :- parent(X,Y).
```

But what if the chain of parents is longer? We could try to add another rule

```
ancestor(X,Y) :- parent(X,Z), parent(Z,Y).
```

Recursive rules (cont)

But what if the chain of ancestors is strictly longer than two parent relationships? We could try to add

```
ancestor(X,Y) :- parent(X,Z1), parent(Z1,Z2), parent(Z2,Y).
```

but we can never attain the general case, when the chain is arbitrarily long... The solution consists in designing a **recursive rule**. For example:

```
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

which translates "`X` is an ancestor of `Y` if `X` is the parent of some `Z` which is in turn the ancestor of `Y`."

Recursive rules (cont)

We also must keep the special case when ancestorship is parenthood, so all together we have

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

This pattern is typical of recursive rules: there is at least one rule which is not recursive, to cover base cases, and some others which are recursive. Let us query for example

```
?- ancestor(pam,X).
X = bob;
X = ann;
X = pat;
X = jim
```