Problem 2: bound variables

We now know how to specify values for free variables: with an assignment. This allowed us to evaluate all quantifier-free formulas. But most formulas involve quantifiers and bound variables.

Values of bound variables are not — and should not be — given by the situation, as they are controlled by quantifiers.

How do we handle this?

Answer:

We let the assignment vary. Rough idea:

- for $\exists$, want some assignment to make the formula true;
- for $\forall$, demand that all assignments make it true.

Semantics of non-atomic formulas (Def. 1.12) ctd.

Notation (not very standard):
Suppose that M is a structure, g, h are assignments into M, and x is a variable.

We write $g =_x h$ if $g(y) = h(y)$ for all variables y other than x.
(Maybe $g(x) = h(x)$ too!)

$g =_x h$ means 'g agrees with h on all variables except possibly x'.

Warning: don't be misled by the '=' sign in $=_x$.
$g =_x h$ does not imply $g = h$, because we may have $g(x) \neq h(x)$.

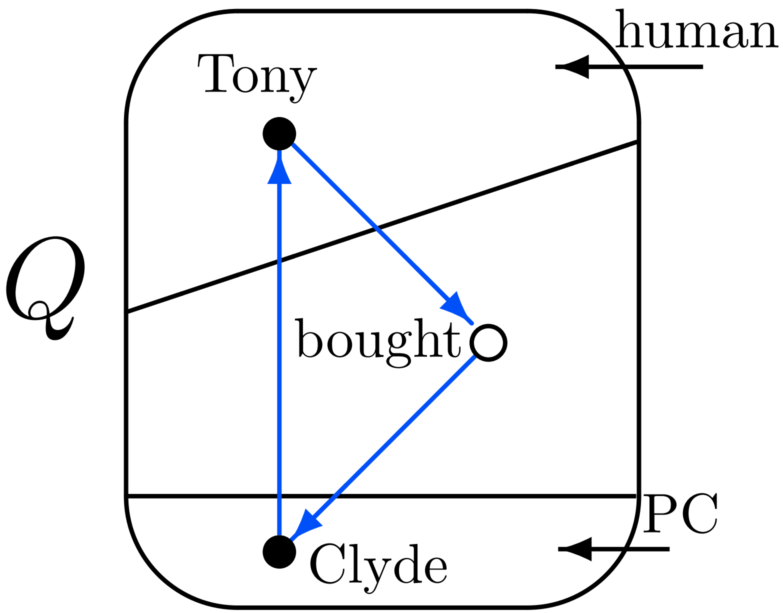
Semantics of non-atomic formulas (Def. 1.12) ctd.

Definition 1.12 (continued)

Suppose we already know how to evaluate the formula A in M under any assignment. Let x be any variable, and h be any assignment into M. Then:

6. $M, h \models \exists x A$ *if* $M, g \models A$ *for some assignment* g *into* M *with* $g =_x h$. *If not, then* $M, h \not\models \exists x A$.
7. $M, h \models \forall x A$ *if* $M, g \models A$ *for every assignment* g *into* M *with* $g =_x h$. *If not, then* $M, h \not\models \forall x A$.

Evaluating formulas with quantifiers: an example



$y \backslash x$	Tony	$\bigcirc$	Clyde	
Tony	h_1	h_2	h_3	$=_x$
$\bigcirc$	h_4	h_5	h_6	$=_x$
Clyde	h_7	h_8	h_9	$=_x$
	$\|_y$	$\|_y$	$\|_y$	

Eg: $h_2(x) = \bigcirc$, and $h_2(y) = \text{Tony}$.

- $Q, h_2 \not\models \texttt{human}(x)$
- $Q, h_2 \models \exists x\, \texttt{human}(x)$, because there is an assignment g with $g =_x h_2$ and $Q, g \models \texttt{human}(x)$ — namely, $g = h_1$
- $Q, h_7 \not\models \forall x\, \texttt{human}(x)$, because it is not true that $Q, g \models \texttt{human}(x)$ for all g with $g =_x h_7$: e.g., $h_8 =_x h_7$ and $Q, h_8 \not\models \texttt{human}(x)$.

A more complex one: $Q, h_4 \models \forall x \exists y, \texttt{bought}(\texttt{x}, \texttt{y})$

For this to be true, we require $Q, g \models \exists y, \texttt{bought}(x, y)$ for every assignment g into Q with $g =_x h_4$.

These are: h_4, h_5, h_6.

- $Q, h_4 \models \exists y\, \texttt{bought}(x, y)$, because
  - $h_4 =_y h_4$ and $Q, h_4 \models \texttt{bought}(x, y)$
- $Q, h_5 \models \exists y\, \texttt{bought}(x, y)$, because
  - $h_8 =_y h_5$ and $Q, h_8 \models \texttt{bought}(x, y)$
- $Q, h_6 \models \exists y\, \texttt{bought}(x, y)$, because
  - $h_3 =_y h_6$ and $Q, h_3 \models \texttt{bought}(x, y)$

So indeed, $Q, h_4 \models \forall x \exists y\, \texttt{bought}(x, y)$.

Useful notation for free variables

The following notation is useful for writing and evaluating formulas. The books often write things like

'Let $A(x_1, \ldots, x_n)$ be a formula.'

This indicates that the free variables of A are among $x_1, \ldots, x_n$.
Note: $x_1, \ldots, x_n$ should all be different. And not all of them need actually occur free in A.
Example: if C is the formula

$$\forall x(R(x, y) \to \exists y S(y, z)),$$

we could write it as

- $C(y, z)$
- $C(x, z, v, y)$
- C (if we're not using the useful notation)

but not as $C(x)$.

Notation for assignments

Fact 1.13

For any formula A, whether or not $M, h \models A$ only depends on $h(x)$ for those variables x that occur free in A.

So for a formula $A(x_1, \ldots, x_n)$, if $h(x_1) = a_1, \ldots, h(x_n) = a_n$, it's OK to write $M \models A(a_1, \ldots, a_n)$ instead of $M, h \models A$.

- Suppose we are explicitly given a formula $C(y, z)$, such as

  $$\forall x(R(x, y) \to \exists y S(y, z)).$$

  If $h(y) = a$, $h(z) = b$, say, we can write

  $$M \models C(a, b), \quad \text{or} \quad M \models \forall x(R(x, a) \to \exists y S(y, b)),$$

  instead of $M, h \models C$. Note: only the *free* occurrences of y in C are replaced by a. The bound y is unchanged.

- For a sentence S, whether $M, h \models S$ does not depend on h at all. So we can just write $M \models S$.

Working out $\models$ in this notation

Suppose we have an L-structure M, an L-formula $A(x, y_1, \ldots, y_n)$, and objects $a_1, \ldots, a_n$ in $\mathrm{dom}(M)$.

- To establish that $M \models (\forall x A)(a_1, \ldots, a_n)$ you check that $M \models A(b, a_1, \ldots, a_n)$ for each object b in $\mathrm{dom}(M)$.
  You have to check even those b with no constants naming them in M. 'Not just Frank, Texel, ..., but all the other ○ and ●
  We can summarise this as a *recursive procedure:*

```
function istrue(M, B) :  bool
... if B = (∀xA)(a1, ..., an) {
    repeat for all b in dom(M):{
      if not istrue(M, A(b, a1, ..., an)) then return false}
  return true
```

The case $\exists x A$, for $A(x, y_1, \ldots, y_n)$

- To establish $M \models (\exists x A)(a_1, \ldots, a_n)$, you try to find some object b in the domain of M such that $M \models A(b, a_1, \ldots, a_n)$.

```
function istrue(M,B) :  bool
...
 if B = (∃xA)(a_1,...,a_n) {
  repeat for all b in dom(M):
    {if istrue(M,A(b,a_1,...,a_n)) then return true}
  return false

 }
```

A is simpler than $\forall x A$ or $\exists x A$. So you can recursively work out if $M \models A(b, a_1, \ldots, a_n)$, in the same way. The process terminates.
Exercise: write the whole function `istrue`. Then implement in Haskell!

So how to evaluate in practice?

We've just seen the formal definition of truth in a structure (due to Alfred Tarski, 1933–1950s).

But how best to work out whether $M \models A$ in practice?

- Often you can do it by working out the English meaning of A and checking it against M.
- Use definition 1.12 and check all assignments. Tedious but can often do mentally with practice. E.g., in $\forall x(\texttt{lecturer}(x) \rightarrow \texttt{PC}(x))$, run through all x and check that every x that's a lecturer is a PC.
- Rewrite the formula in a more understandable form using equivalences (see later).
- Use a combination of the three.

How hard is first-order evaluation?

In most practical cases, with a sentence written by a (sane) human, it's easy to do the evaluation mentally, once used to it.

If the sentence makes no sense, you may have to evaluate it by checking all assignments. Tedious but straightforward.

But in general, evaluation is hard.

It is generally believed that $\forall x \exists y \forall z \exists t \forall u \exists v A$ is just too difficult to understand.

Suppose that N is the structure whose domain is the natural numbers and with the usual meanings of **prime**, **even**, $>$, $+$, 2. No-one knows whether

$$N \models \forall x(\texttt{even}(x) \wedge x > 2 \to \exists y \exists z(\texttt{prime}(y) \wedge \texttt{prime}(z) \wedge x = y + z)).$$