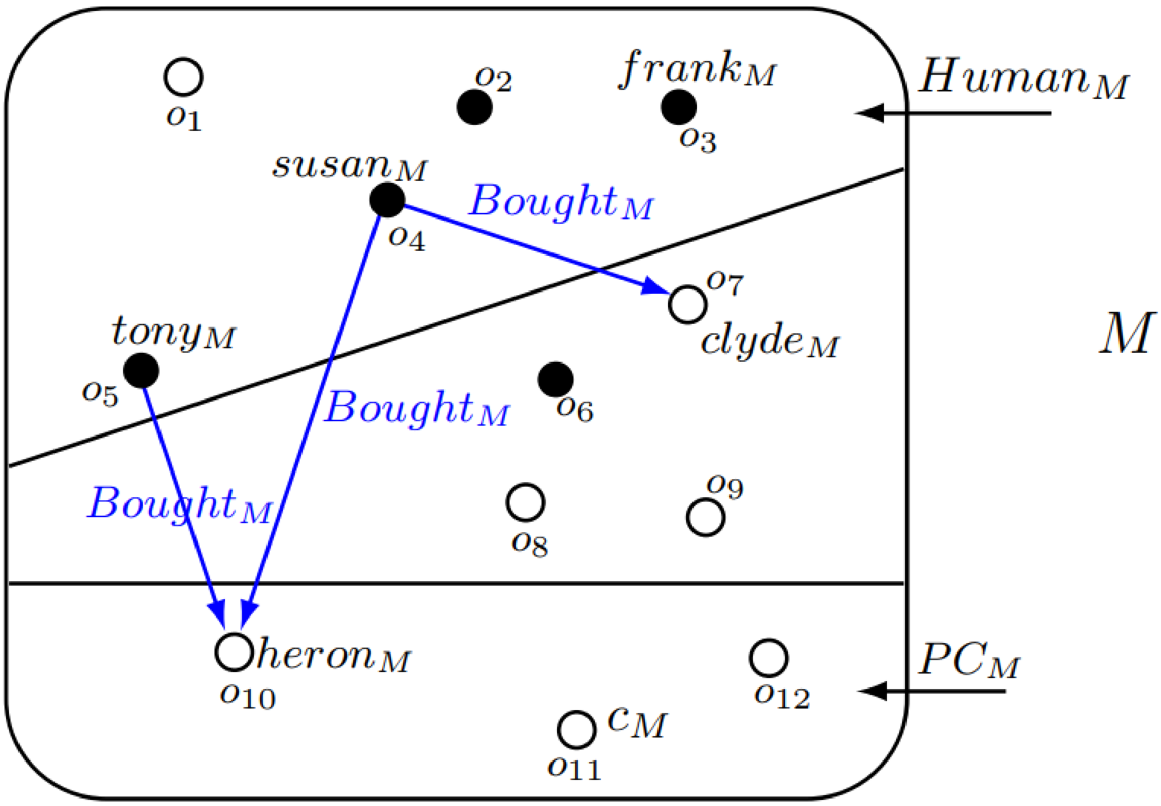
First Order Logic Semantics Part 2



$$M \models \exists x \, \mathtt{Bought}(x, \mathtt{heron}).$$

We can take (for example) x to be o_5, marked on the diagram as $tony_M$.

$$M \models \forall x (\mathtt{Bought}(\mathtt{tony}, x) \to \mathtt{Bought}(\mathtt{susan}, x)).$$

Check those x (here, just the object o_{10}) for which $Bought(tony, x)$ is true (i.e. $(tony_M, heron_M) \in Bought_M$).

For the object $o_{10} = heron_M$, $Bought(susan, heron)$ is true in M : $(susan_M, heron_M) \in Bought_M$

Therefore,

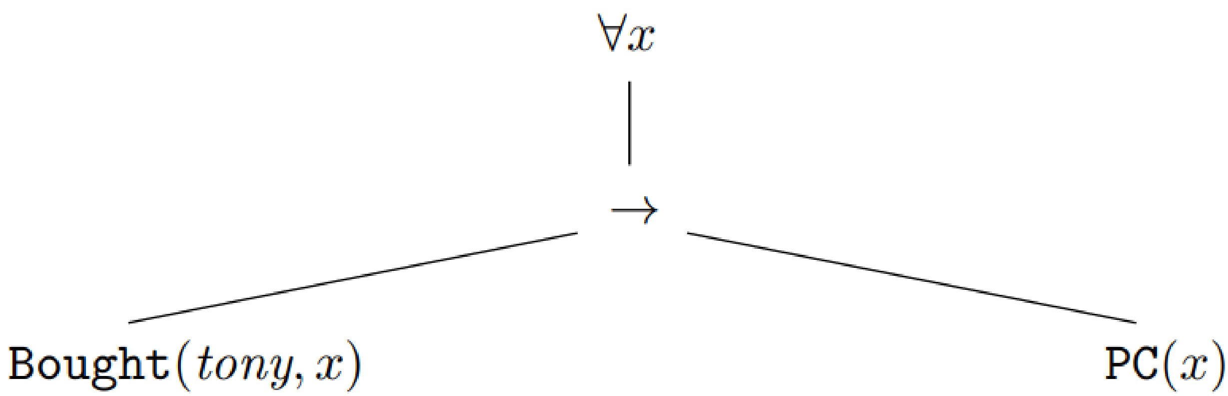
$$M \models \forall x(\mathtt{Bought}(\mathtt{tony}, x) \to \mathtt{Bought}(\mathtt{susan}, x)).$$

The effect of '$\forall x(\mathtt{Bought}(\mathtt{tony}, x) \to \cdots$' is to *restrict the* $\forall x$ to those x that Tony bought. *This trick is extremely useful. Remember it!*

Consider:

$$\forall x(\mathtt{Bought}(\mathtt{tony}, x) \to \mathtt{PC}(x))$$

Its formation tree is:



We cannot evaluate the parity of the main formula by working up the tree, because the parities of the leaves depend on the value of x.

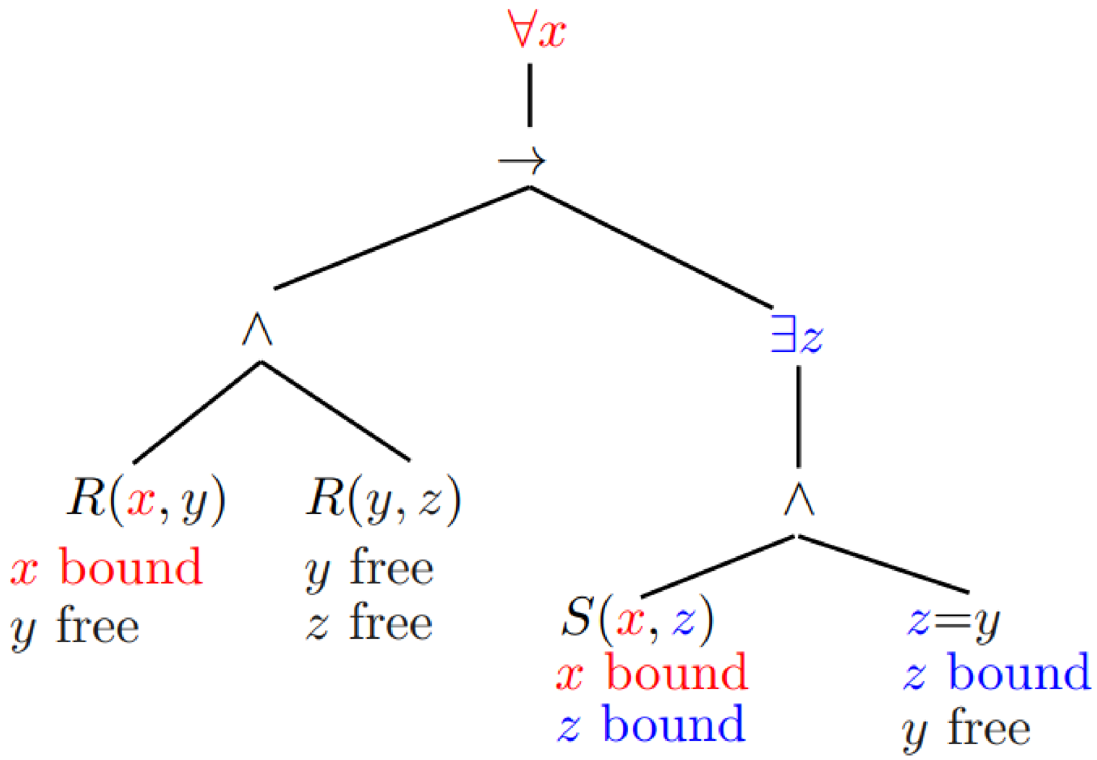
Definition 4.5 (free and bound variables)

Let ϕ be a formula.

1. An occurrence of a variable x in ϕ is said to be *bound* if it occurs *in the scope of a quantifier* $\forall x$ or $\exists x$.
2. Variables that are not bound are said to be *free.*
3. The *free variables of* ϕ are those variables with free occurrences in ϕ.

A variable x that is bound in ϕ occurs in an atomic subformula of ϕ that lies under a quantifier $\forall$ x or $\exists$ x in the formation tree of ϕ.

$$\forall x(R(x,y) \land R(y,z) \rightarrow \exists z(S(x,z) \land z = y))$$



The free variables of the formula are y and z. Note: z has both free and bound occurrences.

A formulae with free variables is neither true nor false in a structure M, because the variables have no meaning in M. We must always specify values for free variables using an assignment. What a structure does for constants, an assignment does for variables.

Definition 4.6 (assignment)

Let $M = \langle \mathbb{D}, \mathbb{I} \rangle$ be a structure. An *assignment (or 'valuation') over M* is a function that assigns an object in $\mathbb{D}$ to each variable. That is, $h : V \mapsto \mathbb{D}$ is an assignment, where V is the set of variables.

For an assignment h and a variable x, we write $h(x)$ to denote the object in $\mathbb{D}$ assigned to x by h.

An L-structure M plus an assignment h over M form a 'complete situation'. We can then evaluate:

- any L-term to an object in dom(M)

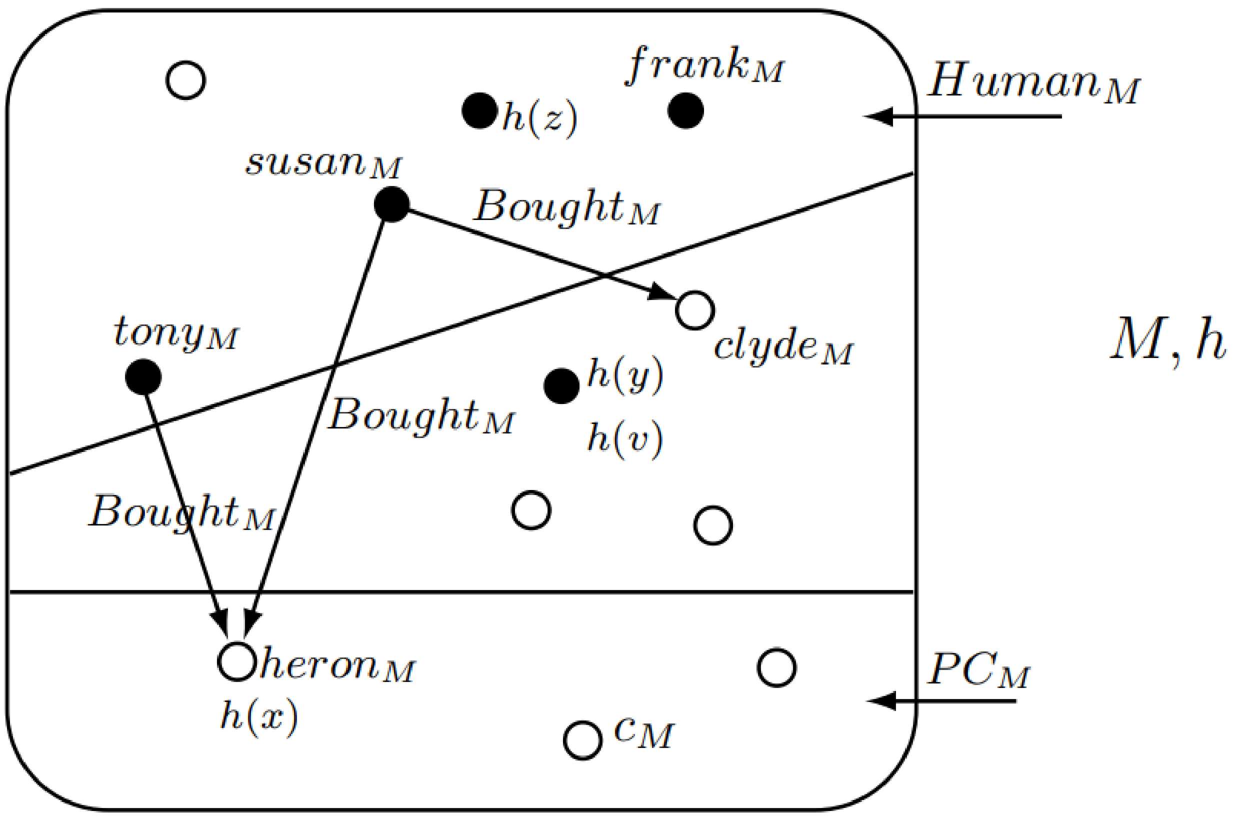
- any L-formula with no quantifiers to true or false

We do the evaluation in two stages: first terms, then formulas.

Definition 4.7 (value of a term)

Let L be a signature, $M = \langle \mathbb{D}, \mathbb{I} \rangle$ an L-structure, and h an assignment over M. Then for any L-term t, the *value of t in M under h*, denoted as $|t|_M^h$, is the object in $\mathbb{D}$ allocated to t by:

- M, if t is a constant — that is, $|t|_M^h = \mathbb{I}(t) = t_M$
- h, if t is a variable — that is, $|t|_M^h = h(t)$.
- M and h, if t is a term $f(t_1, \ldots, t_n)$ — that is, $|t|_M^h = f_M(|t_1|_M^h, \ldots, |t_n|_M^h)$



A useful signature for arithmetic and for programs using numbers is the L consisting of:

- constants $\underline{\mathtt{0}}$, $\underline{\mathtt{1}}$, $\underline{\mathtt{2}}$, … (I use underlined typewriter font to avoid confusion with actual numbers $0, 1, \ldots$)
- binary function symbols $+, -, \times$
- binary relation symbols $<, \leq, >, \geq$.

We'll abuse notation by writing L-terms and formulas in infix notation (everybody does this, but it breaks Definitions 4.2 and 4.3):

- $x + y$, rather than $+(x, y)$,
- $x > y$, rather than $>(x, y)$.

Examples of terms: $x + \underline{\mathtt{1}}$, $\quad \underline{\mathtt{2}} + (x + \underline{\mathtt{5}})$, $\quad (\underline{\mathtt{3}} \times \underline{\mathtt{7}}) + x$. Not $x + y + z$.
Examples of formulas: $\underline{\mathtt{3}} \times x > \underline{\mathtt{0}}$, $\quad \forall x(x > \underline{\mathtt{0}} \to x \times x > x)$.

We evaluate arithmetic terms in a structure with domain $\mathbb{D} = \{0, 1, 2, \ldots\}$ in the obvious way.

But (eg) $34 - 61$ is unpredictable — can be any number.

We can now evaluate any formula without quantifiers.
Fix an L-structure M and an assignment h.

We write $M, h \models \phi$ if ϕ is true in M under h, and $M, h \not\models \phi$ if not.

We can now evaluate any formula without quantifiers.

Fix an L-structure M and an assignment h.

We write $M, h \models \phi$ if ϕ is true in M under h, and $M, h \not\models \phi$ if not

Definition 4.8

1. Let R be an n-ary predicate symbol in L, and $t_1, \ldots, t_n$ be L-terms (see Def. 4.2). Let $|t_i|_M^h = a_i$ be the value of t_i in M under h for each $i = 1, \ldots, n$.
   $M, h \models R(t_1, \ldots, t_n)$ if $(a_1, \ldots, a_n) \in R_M$. If not, then $M, h \not\models R(t_1, \ldots, t_n)$.
2. Let t, t' be terms. Then
   $M, h \models t = t'$ if t and t' have the same value in M under h, that is $|t|_M^h = |t'|_M^h$. If they don't, then $M, h \not\models t = t'$.
3. $M, h \models \top$, and $M, h \not\models \bot$.
4. $M, h \models A \wedge B$ if $M, h \models A$ and $M, h \models B$. Otherwise, $M, h \not\models A \wedge B$.
5. $\neg A$, $A \vee B$, $A \to B$, $A \leftrightarrow B$ — as in propositional logic.

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 complete situation, as they are controlled by quantifiers.

How do we handle this?

Answer: Informally, we let the assignment vary. Rough idea:

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

Formally, we use the notion of *[variable]-equivalent* variable assignments.

Two variable assignments are [*variable*]-equivalent if they differ at most in the assignment of the variable "[*variable*]".

Let M be a structure, g, h be two assignments under M, and x be a variable.

We say that *g and h are x-equivalent*, written $g =_x h$, if they differ at most in the assignment of x.

- The following four variable assignments are y-equivalent.
  - h_1: $h_1(x) = a_1,\ h_1(y) = a_2,\ h_1(z) = a_3$
  - h_2: $h_2(x) = a_1,\ h_2(y) = a_4,\ h_2(z) = a_3$
  - h_3: $h_3(x) = a_1,\ h_3(y) = a_6,\ h_3(z) = a_3$
  - h_4: $h_4(x) = a_1,\ h_4(y) = a_2,\ h_4(z) = a_3$

Note: A variable assignment is always [*variable*]-equivalent to itself.

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

Definition 4.9 (Def. 4.8 continued)

Let M be a L-structure and h be any assignment over M.

Suppose we already know how to evaluate a formula ϕ in M under any assignment. Let x be any variable. Then:

6. $M, h \models \exists x\phi$ if $M, g \models \phi$ for *some* assignment g over M that is $g =_x h$. If not, then $M, h \not\models \exists x\phi$.
7. $M, h \models \forall x\phi$ if $M, g \models \phi$ for *every* assignment g over M that is $g =_x h$. If not, then $M, h \not\models \forall x\phi$.

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

For this to be true, we require $Q, g \models \exists y, \texttt{Bought}(x, y)$ for every assignment g over 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, $Q, h_4 \models \forall x \exists y\, \texttt{Bought}(x, y)$.

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

This indicates that the free variables of ϕ are among $x_1, \ldots, x_n$.

$x_1, \ldots, x_n$ should all be different.
Not all of them need be free in ϕ

Example: if ϕ is the formula

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

we could write it as

- $\phi(y,z)$
- $\phi(x,z,y)$
- ϕ (if we're not using the useful notation)

but not as $\phi(x)$.

Fact 1

Given a formula ϕ, whether or not $M, h \models \phi$ only depends on $\phi(x)$ for those variables x that occur free in ϕ.

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

No free variables (a sentence)? Forget the h.

Definition 4.10 (sentence)

A *sentence* is a formula with no free variables.

Proving $M \models \phi$

- Convert to English
- Check all assignments
- Rewrite the formula into a more understandable form
- Use a combination of the three

Example

$$M \models \forall x(\texttt{Lecturer}(x) \to \texttt{Human}(x))$$

Use Notation along these lines:

$$|x|^h_M = o_6. \text{ But } o_6 \notin \mathbb{I}(\texttt{Human}).$$