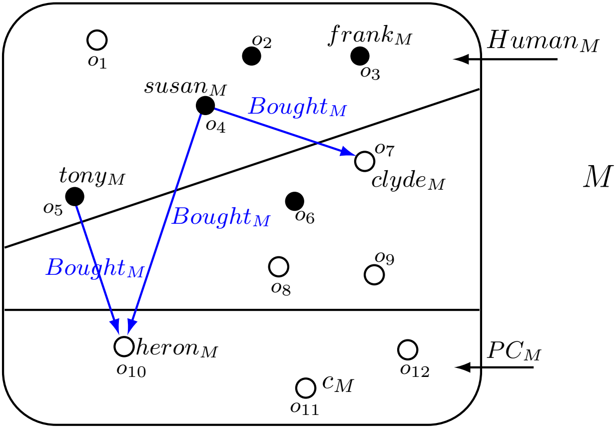
Discrete mathematics, **logic** and reasoning (COMP40018)

Dalal Alrajeh

Many thanks to Ian Hodkinson and Robert Craven for some of the original material, and none of the errors.

Evaluating quantifiers — rough guide

Let's go back to our L-structure M.



Evaluating quantifiers — rough guide

How can we tell if $\exists x\ \texttt{Bought}(x, \texttt{heron})$ is true in M?
In symbols, do we have $M \models \exists x\ \texttt{Bought}(x, \texttt{heron})$?
In English, 'does M say that something bought Heron?'.

Evaluating quantifiers — rough guide

How can we tell if $\exists x\ \texttt{Bought}(x, \texttt{heron})$ is true in M?
In symbols, do we have $M \models \exists x\ \texttt{Bought}(x, \texttt{heron})$?
In English, 'does M say that something bought Heron?'.

Well, for this to be so, there must be an object x in $\mathbb{D}$ such that $M \models \texttt{Bought}(x, \texttt{heron})$ — that is, M says that x `Bought` `heron`.

Evaluating quantifiers — rough guide

How can we tell if $\exists x\ \texttt{Bought}(x, \texttt{heron})$ is true in M?
In symbols, do we have $M \models \exists x\ \texttt{Bought}(x, \texttt{heron})$?
In English, ‘does M say that something bought Heron?’.

Well, for this to be so, there must be an object x in $\mathbb{D}$ such that $M \models \texttt{Bought}(x, \texttt{heron})$ — that is, M says that x `Bought heron`.

There is: we have a look, and we see that we can take (eg.) x to be o_5 marked on the diagram as $tony_M$.

So yes indeed, $M \models \exists x\ \texttt{Bought}(x, \texttt{heron})$.

Another example: $M \models \forall x(\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x))$?

Is it true that "for every object x in $\mathbb{D}$,

$\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x)$ is true in M"?

Another example: $M \models \forall x(\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x))$?

Is it true that "for every object x in $\mathbb{D}$,
$\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x)$ is true in M"?

In M, there are 12 possible x.

We need to check whether $\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x)$ is true in M for each of the 12 possible x in M.

Another example: $M \models \forall x(\texttt{Bought}(\texttt{tony}, x) \rightarrow \texttt{Bought}(\texttt{susan}, x))$?

Is it true that "for every object x in $\mathbb{D}$,
$\texttt{Bought}(\texttt{tony}, x) \rightarrow \texttt{Bought}(\texttt{susan}, x)$ is true in M"?

In M, there are 12 possible x.

We need to check whether $\texttt{Bought}(\texttt{tony}, x) \rightarrow \texttt{Bought}(\texttt{susan}, x)$ is true in M for each of the 12 possible x in M.

BUT

$\texttt{Bought}(\texttt{tony}, x) \rightarrow \texttt{Bought}(\texttt{susan}, x)$ true in M for any object x such that $\texttt{Bought}(\texttt{tony}, x)$ is false in M.
('False $\rightarrow$ anything is true.')

Another example: $M \models \forall x(\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x))$?

Is it true that "for every object x in $\mathbb{D}$,
$\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x)$ is true in M"?

In M, there are 12 possible x.

We need to check whether $\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x)$ is true in M for each of the 12 possible x in M.

BUT

$\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x)$ true in M for any object x such that $\texttt{Bought}(\texttt{tony}, x)$ is false in M.
('False $\to$ anything is true.')

So we only need check those x — here, just the object o_{10} — for which $\texttt{Bought}(\texttt{tony}, x)$ is true, (i.e. $(tony_M, heron_M) \in Bought_M$).

For the object $o_{10} = heron_M$,

$$\texttt{Bought}(\texttt{susan}, \texttt{heron}) \text{ is true in } M$$
$$(susan_M, heron_M) \in Bought_M$$

So $\quad\texttt{Bought}(\texttt{tony}, x) \to \texttt{Bought}(\texttt{susan}, x)$ is true in M
for *every* object x in M

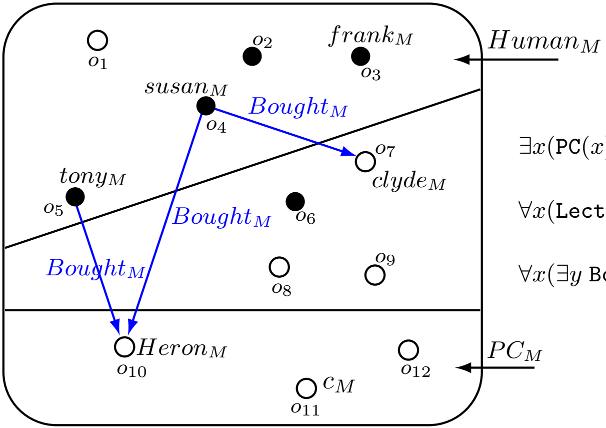
Hence,

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

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

Exercise: which are true in M?

Remember: the black dots are the lecturers.



$\exists x(\texttt{PC}(x) \wedge \texttt{Bought}(\texttt{frank}, x))$

$\forall x(\texttt{Lecturer}(x) \rightarrow \texttt{Human}(x))$

$\forall x(\exists y\ \texttt{Bought}(y, x) \rightarrow \neg\texttt{Human}(x))$

Truth in a structure — formally!

We saw informally how to evaluate formulas with no variables and formulas where variables are quantified, in a structure 'by inspection'.

But as in propositional logic, English can only be a rough guide. For engineering, this is not good enough.

We need a more formal way to evaluate all predicate logic formulas in structures.

In propositional logic, we calculated the truth value of a formula in a situation by working up through its formation tree — from the atomic subformulas (leaves) up to the root.

For predicate logic, things are not so simple...

A problem

Consider the formula $\forall x(\texttt{Bought}(\texttt{tony}, x) \to \texttt{PC}(x))$ in the structure M given in Slide 183. Its formation tree is:

$$
\begin{tikzcd}
 & \forall x \arrow[d, dash] & \\
 & \to \arrow[dl, dash] \arrow[dr, dash] & \\
\texttt{Bought}(tony, x) & & \texttt{PC}(x)
\end{tikzcd}
$$

Can we evaluate the main formula by working up the tree?

Is $\texttt{Bought}(\texttt{tony}, x)$ true in M?!

Is $\texttt{PC}(x)$ true in M?!

It depends on what x is! So, what's going on?

Free and bound variables

We'd better investigate how variables can arise in formulas.

Definition 4.5 (free and bound variables)

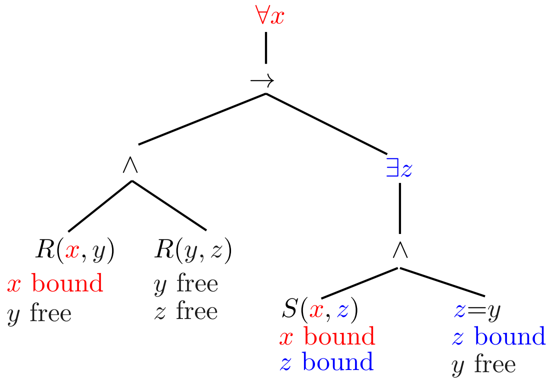
Let ϕ be a formula.

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 variables 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.

Example

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



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

Problem 1: free variables

A formulae with free variables is neither true nor false in a structure M, because the variables have no meaning in M.

It's like asking 'is $x = 7$ true?'

So, *the structure is not a 'complete' situation* — it doesn't fix the meanings of free variables. (They are *variables,* after all!)

So we must specify values for free variables, before evaluating a formula to true or false.

This is so even if it turns out that the values do not affect the answer (like $x = x$).

Assignments to variables

We supply the missing values of free variables using something called 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.

Evaluating terms

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*.

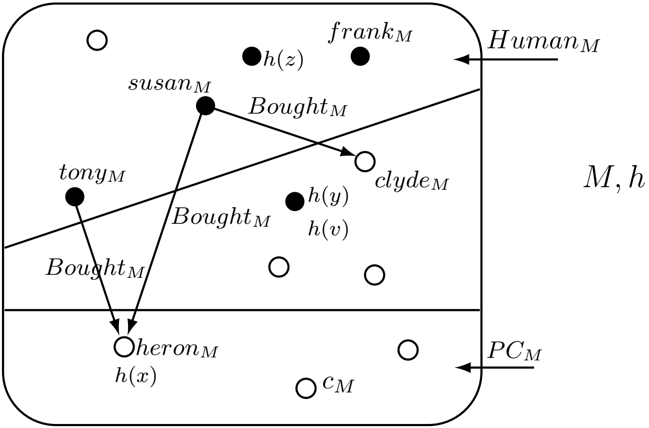
We do the evaluation in two stages: first terms, then formulas.

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, $|t|_M^h = f_M(|t_1|_M^h, \ldots, |t_n|_M^h)$

Evaluating terms: example



The value in M under h of the term `tony` is:

The value in M under h of the term x is:

Example: evaluating arithmetic terms

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

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

Example: evaluating arithmetic terms

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 $>(x, y)$.

Example: evaluating arithmetic terms

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: $\underline{\mathtt{3}} \times x > \underline{\mathtt{0}}$, $\quad \forall x(x > \underline{\mathtt{0}} \to x \times x > x)$.

Example: evaluating arithmetic terms

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.

Semantics of quantifier-free formulas

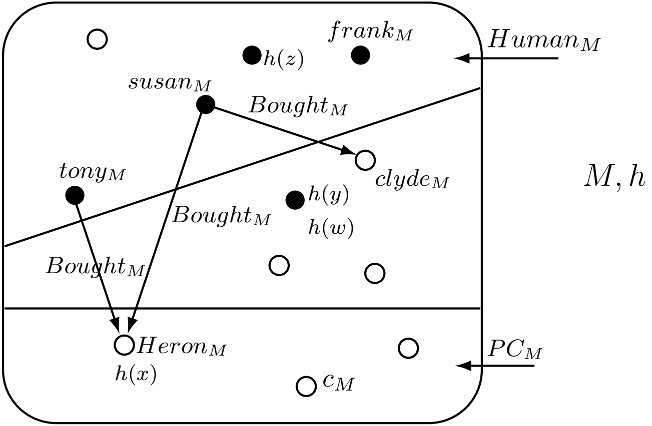
We can now evaluate any formula without quantifiers.
Fix an L-structure M and an assignment h.

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.

Evaluating quantifier-free formulas: example



- $M, h \models \texttt{Human}(z)$?
- $M, h \models x = \texttt{heron}$?
- $M, h \models \texttt{Bought}(\texttt{susan}, y) \vee z = \texttt{frank}$?

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 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.

[*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: $\quad h_1(x) = a_1,\ h_1(y) = a_2,\ h_1(z) = a_3$
  - h_2: $\quad h_2(x) = a_1,\ h_2(y) = a_4,\ h_2(z) = a_3$
  - h_3: $\quad h_3(x) = a_1,\ h_3(y) = a_6,\ h_3(z) = a_3$
  - h_4: $\quad h_4(x) = a_1,\ h_4(y) = a_2,\ h_4(z) = a_3$

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 $g(x) \neq h(x)$.

Semantics of quantified formulas

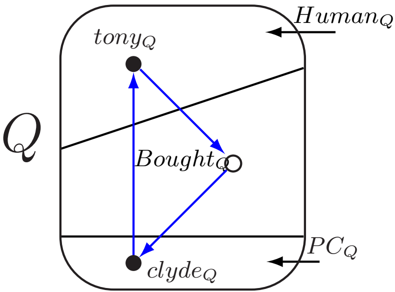
Definition 4.9 (Def. 4.8 continued)

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 $M, h \not\models \forall x \phi$.

Evaluating formulas with quantifiers: an example



$y \backslash x$	$tony_Q$	$\bigcirc$	$clyde_Q$	
$tony_Q$	h_1	h_2	h_3	$=_x$
$\bigcirc$	h_4	h_5	h_6	$=_x$
$clyde_Q$	h_7	h_8	h_9	$=_x$
	$=_y$	$=_y$	$=_y$	

Eg: $h_2(x) = \bigcirc$, and $h_2(y) = tony_Q$

- $Q, h_2 \not\models \texttt{Human}(x)$
- $Q, h_2 \models \exists x\ \texttt{Human}(x)$, because there is an assignment h_1 where $h_1 =_x h_2$ and $Q, h_1 \models \texttt{Human}(x)$.
- $Q, h_7 \not\models \forall x\ \texttt{Human}(x)$, 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 $Q, h_8 \not\models \texttt{Human}(x)$.

The object assigned to y is irrelevant here. But it is needed next...

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

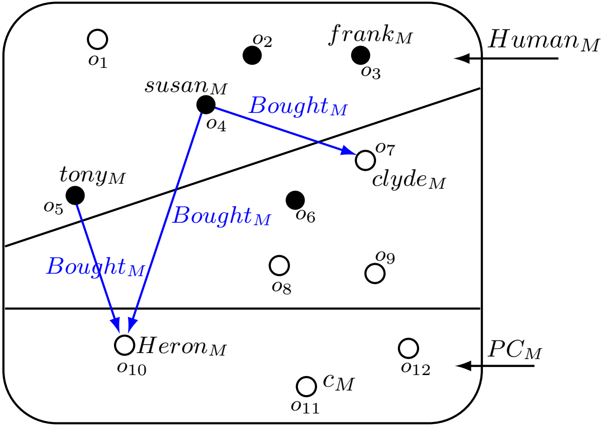
For this to be true, we require $Q, g \models \exists y, \texttt{Bought}(x, y)$ for every assignment g over Q with $g =_x h_4$.

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

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



Let h be arbitrary assignment. There are two cases. Take $|x|_M^h \in \mathbb{I}(\texttt{Lecturer})$. That is $|x|_M^h$ is either o_3, o_2, o_4, o_5, o_6. Take h to be an assignment where $|x|_M^h = o_6$. But $o_6 \notin \mathbb{I}(\texttt{Human})$. Hence $M, h \not\models \texttt{Human}(x)$. Therefore $M, h \not\models (\texttt{Lecturer}(x) \to \texttt{Human}(x))$. Hence $M \not\models \forall x(\texttt{Lecturer}(x) \to \texttt{Human}(x))$

Useful notation for free variables

The following notation is useful for writing and evaluating formulas.

The books often write things like

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

This indicates that the free variables of ϕ are among $x_1, \ldots, x_n$. Note: x_1, $\ldots, x_n$ should all be different. And not all of them need actually occur 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 $\phi(x)$.

Notation for assignments

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 $M, h \models \phi$.

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

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

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

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

  instead of $M, h \models \phi$. Note: only the *free* occurrences of y in ϕ 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 $M \models S$.

Sentences

Definition 4.10 (sentence)

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

Example:

- $\forall x(\texttt{Bought}(\texttt{tony}, x) \to \texttt{PC}(x))$ is a sentence.
- Its subformulas are not sentences:

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

Which are sentences?

- $\texttt{Bought}(\texttt{frank}, \texttt{heron})$
- $x = x$
- $\texttt{Bought}(\texttt{susan}, x)$
- $\forall x(\exists y(y = x) \to x = y)$
- $\forall x \forall y(x = y \to \forall z(\texttt{R}(x, z) \to \texttt{R}(y, z)))$

Useful notation for free variables

The following notation is useful for writing and evaluating formulas.

The books often write things like

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

This indicates that the free variables of ϕ are among $x_1, \ldots, x_n$. Note: x_1, $\ldots, x_n$ should all be different. And not all of them need actually occur 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 $\phi(x)$.