1.2 Semantics

The notion of situations has to be extended to give meaning to the new style of predicate logic formulas. We have to specify

- what a *situation* is for predicate logic,
- how to evaluate predicate logic formulas in a given situation.

We have to handle new atomic formulas, quantifiers and variables.

Structures (i.e. situations in predicate logic)

Let's deal with the new-style atomic formulas first.

Definition 1.5 (structure)

Let L be a signature. An L-structure (or sometimes (loosely) a model) M is a thing that

- identifies a non-empty collection (set) of objects that M 'knows about'. It's called the domain or universe of M, written $\mathrm{dom}(M)$.
- specifies what the symbols of L mean in terms of these objects.

The interpretation in M of a constant is an object in $\mathrm{dom}(M)$.

The interpretation in M of a relation symbol is a relation on $\mathrm{dom}(M)$.

CS1 have already seen sets and relations in Discrete Structures.

Example of a structure

For our simple signature L, an L-structure should say:

- which objects are in its domain
- which of its objects are `Frank`, `Susan`, ...
- which objects are `human`, PC, `lecturer`
- which objects `bought` which.

Next slide shows a diagram of an L-structure, called M.

- There are 12 objects (the 12 dots) in dom(M).
- Some objects are labelled (eg 'Frank') to show the meanings of the constants of L (eg `Frank`).
- The interpretations (meanings) of PC, `human` are drawn as regions. The interpretation of `lecturer` is indicated by the black dots.
- The interpretation of `bought` is shown by the arrows between objects.

Example of a structure

For our simple signature L, an L-structure should say:

- which objects are in its domain
- which of its objects are `Frank`, `Susan`, ...
- which objects are `human`, PC, `lecturer`
- which objects `bought` which.

Next slide shows a diagram of an L-structure, called M.

- There are 12 objects (the 12 dots) in dom(M).
- Some objects are labelled (eg 'Frank') to show the meanings of the constants of L (eg `Frank`).
- The interpretations (meanings) of PC, `human` are drawn as regions. The interpretation of `lecturer` is indicated by the black dots.
- The interpretation of `bought` is shown by the arrows between objects.

Example of a structure

For our simple signature L, an L-structure should say:

- which objects are in its domain
- which of its objects are `Frank`, `Susan`, ...
- which objects are `human`, `PC`, `lecturer`
- which objects `bought` which.

Next slide shows a diagram of an L-structure, called M.

- There are 12 objects (the 12 dots) in $\text{dom}(M)$.
- Some objects are labelled (eg 'Frank') to show the meanings of the constants of L (eg `Frank`).
- The interpretations (meanings) of `PC`, `human` are drawn as regions. The interpretation of `lecturer` is indicated by the black dots.
- The interpretation of `bought` is shown by the arrows between objects.

Example of a structure

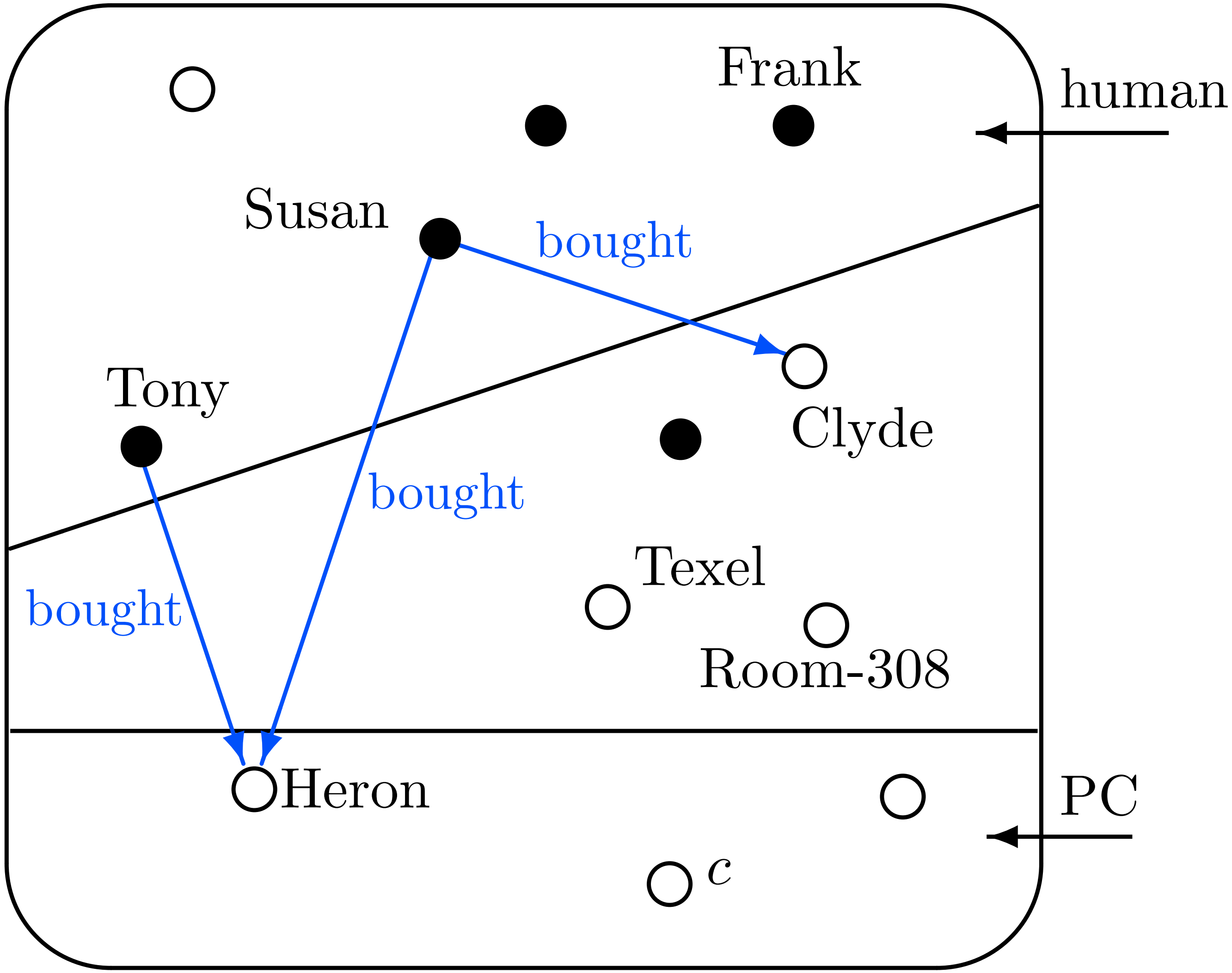
For our simple signature L, an L-structure should say:

- which objects are in its domain
- which of its objects are `Frank`, `Susan`, ...
- which objects are `human`, `PC`, `lecturer`
- which objects `bought` which.

Next slide shows a diagram of an L-structure, called M.

- There are 12 objects (the 12 dots) in $\mathrm{dom}(M)$.
- Some objects are labelled (eg 'Frank') to show the meanings of the constants of L (eg `Frank`).
- The interpretations (meanings) of `PC`, `human` are drawn as regions. The interpretation of `lecturer` is indicated by the black dots.
- The interpretation of `bought` is shown by the arrows between objects.

The structure M



Tony or `Tony`?

Object ● marked ‘Tony’ in $\mathrm{dom}(M)$ and constant `Tony` in L are two different things (Note the different fonts.)

`Tony` is syntactic. ● is semantic.

In M `Tony` is a name for the object ● marked ‘Tony’.

Notation 1.6

Let M be an L-structure and c a constant in L. We write c^M for the interpretation of c in M. It is the object in $\mathrm{dom}(M)$ that c names in M.

$\mathtt{Tony}^M =$ the object ● marked ‘Tony’.

We will usually write ‘Tony’ or $\mathtt{Tony}^M$ (but NOT `Tony`) for this ●

Hence, the meaning of a constant c IS the object c^M assigned to it by a structure M. A constant (and any symbol of L) has as many meanings as there are L-structures.

Alessandra Russo
C140 Logic - 1st year course

Tony or `Tony`?

Object ● marked ‘Tony’ in dom(M) and constant `Tony` in L are two different things (Note the different fonts.)

`Tony` is syntactic. ● is semantic.

In M `Tony` is a name for the object ● marked ‘Tony’.

Notation 1.6

Let M be an L-structure and c a constant in L. We write c^M for the interpretation of c in M. It is the object in dom(M) *that c names in M.*

$$\texttt{Tony}^M = \text{ the object ● marked ‘Tony’.}$$

We will usually write ‘Tony’ or $\texttt{Tony}^M$ (but NOT `Tony`) for this ●

Hence, the meaning of a constant c IS the object c^M assigned to it by a structure M. A constant (and any symbol of L) has as many meanings as there are L-structures.

Tony or `Tony`?

Object ● marked 'Tony' in dom(M) and constant `Tony` in L are two different things (Note the different fonts.)

`Tony` is syntactic. ● is semantic.

In M `Tony` is a name for the object ● marked 'Tony'.

Notation 1.6

Let M be an L-structure and c a constant in L. We write c^M for the interpretation of c in M. It is the object in dom(M) *that c names in M.*

$$\texttt{Tony}^M = \text{ the object ● marked ‘Tony’.}$$

We will usually write 'Tony' or $\texttt{Tony}^M$ (but NOT `Tony`) for this ●

Hence, the meaning of a constant c IS the object c^M assigned to it by a structure M. A constant (and any symbol of L) has as many meanings as there are L-structures.

Drawing other symbols

Our simple signature L has only constants and unary and binary relation symbols.

For this L, we drew an L-structure M by

- drawing a collection of objects (the domain of M)
- marking which objects are named by which constants in M
- marking which objects M says satisfy the unary relation symbols (`human`, etc)
- drawing arrows between the objects that M says satisfy the binary relation symbols. The arrow direction matters.

With several binary relation symbols in L, we'd really need to label the arrows.

It is difficult to draw interpretations of 3-ary or higher-arity relation symbols.

0-ary relation symbols are the same as propositional atoms.

Truth in a structure(a rough guide)

When is a formula without quantifiers true in a structure?

- $\texttt{PC}(\texttt{Heron})$ is true in M, because $\texttt{Heron}^M$ is an object ○ that M says is a $\texttt{PC}$.

  We write this as $M \models \texttt{PC}(\texttt{Heron})$. Read as '$M$ says $\texttt{PC}(\texttt{Heron})$'.

  Warning: This is a quite different use of $\models$ from what seen in the definition of valid argument. '$\models$' is *overloaded* — it's used for two different things.

- $\texttt{bought}(\texttt{Susan},\texttt{Susan})$ is false in M, because M does not say that the constant $\texttt{Susan}$ names an object ● that bought itself. We write this as $M \not\models \texttt{bought}(\texttt{Susan}, \texttt{Susan})$.

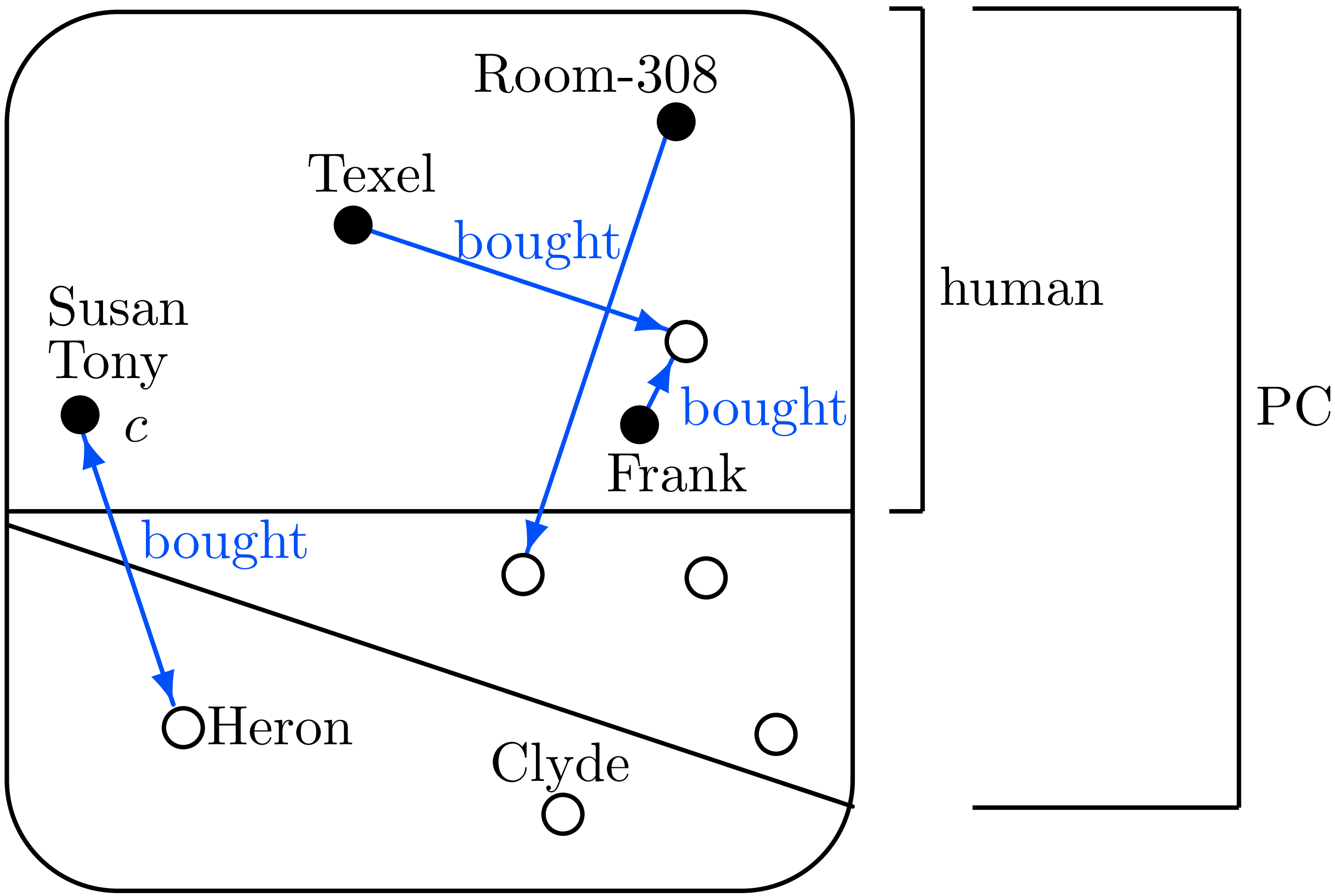
From our knowledge of propositional logic,

- $M \models \neg\, \texttt{human}(\texttt{Room-308})$,
- $M \not\models \texttt{PC}(\texttt{Tony}) \vee \texttt{bought}(\texttt{Frank}, \texttt{Clyde})$.

Alessandra Russo
C140 Logic - 1st year course

Another structure

Here's another L-structure, called M'.



Now, there are only 10 objects in $\mathrm{dom}(M')$.

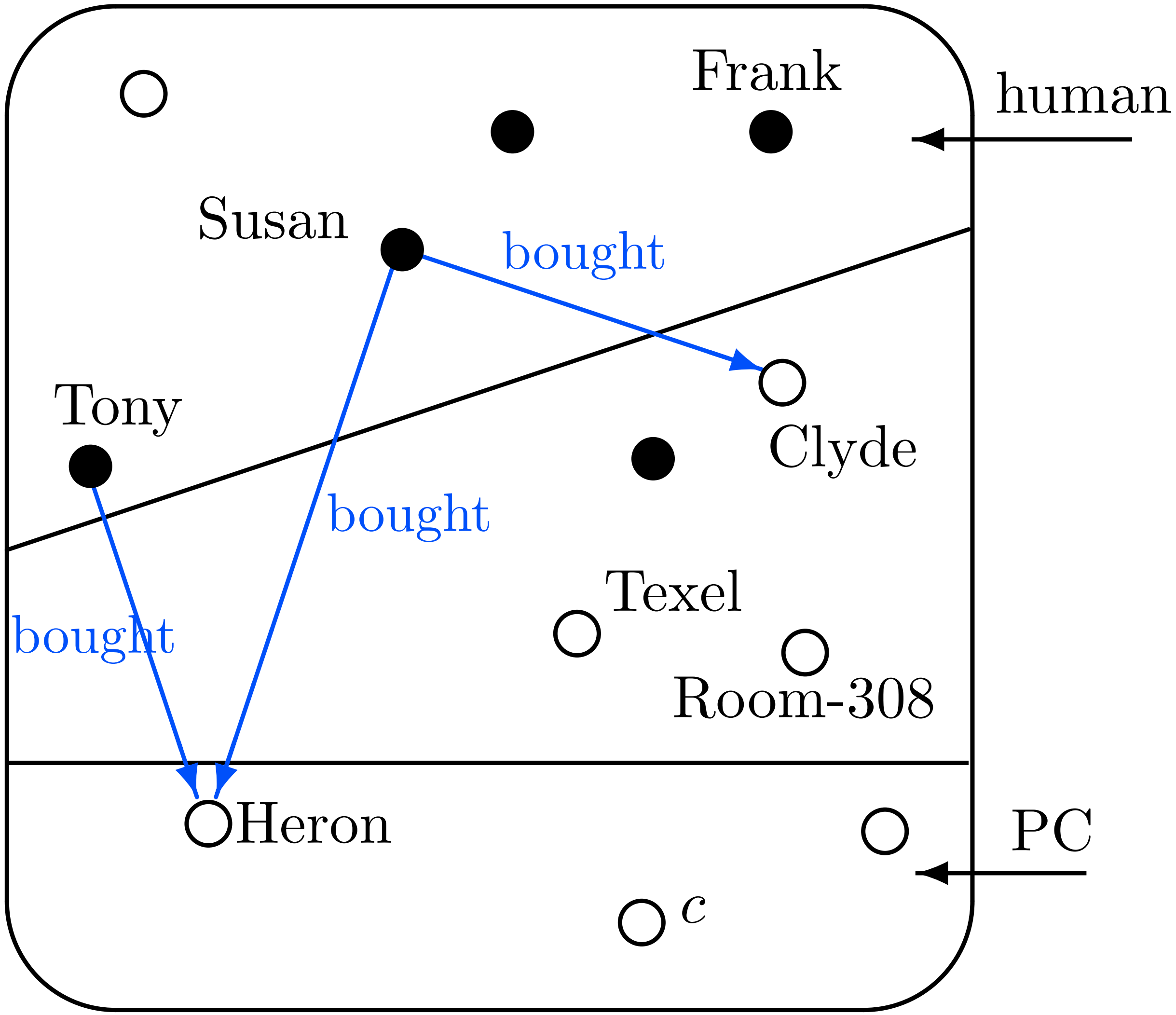
Some statements about M'

- $M' \not\models \texttt{bought}(\texttt{Susan}, \texttt{Clyde})$ this time.
- $M' \models \texttt{Susan} = \texttt{Tony}$.
- $M' \models \texttt{human}(\texttt{Texel}) \wedge \texttt{PC}(\texttt{Texel})$.
- $M' \models \texttt{bought}(\texttt{Tony}, \texttt{Heron}) \wedge \texttt{bought}(\texttt{Heron}, \texttt{c})$.

How about

- $\texttt{bought}(\texttt{Susan}, \texttt{Clyde}) \to \texttt{human}(\texttt{Clyde})$?
- $\texttt{bought}(c, \texttt{Heron}) \to \texttt{PC}(\texttt{Clyde}) \vee \neg\texttt{human}(\texttt{Texel})$?

Evaluating formulas with quantifiers — rough guide



M again

Evaluating quantifiers

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 $\mathrm{dom}(M)$ such that $M \models \texttt{bought}(x, \texttt{Heron})$ — that is, M says that x $\texttt{bought}$ $\texttt{Heron}^M$.

There is: we have a look, and we see that we can take (eg.) x to be (the ● marked) Tony.

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 dom(M),

$$\texttt{bought}(\texttt{Tony}, x) \to \texttt{bought}(\texttt{Susan}, x) \text{ is true in } M\text{"?}$$

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 $\bigcirc = \texttt{Heron}^M$ — for which $\texttt{bought}(\texttt{Tony}, x)$ is true.

For the object $\bigcirc = \texttt{Heron}^M$,

$$\texttt{bought}(\texttt{Susan}, \bigcirc) \text{ is true in } M.$$

So $\quad \texttt{bought}(\texttt{Tony}, \bigcirc) \to \texttt{bought}(\texttt{Susan}, \bigcirc)$ is true in M.

So $\quad \texttt{bought}(\texttt{Tony}, x) \to \texttt{bought}(\texttt{Susan}, x)$ is true in M for *every* object x in M

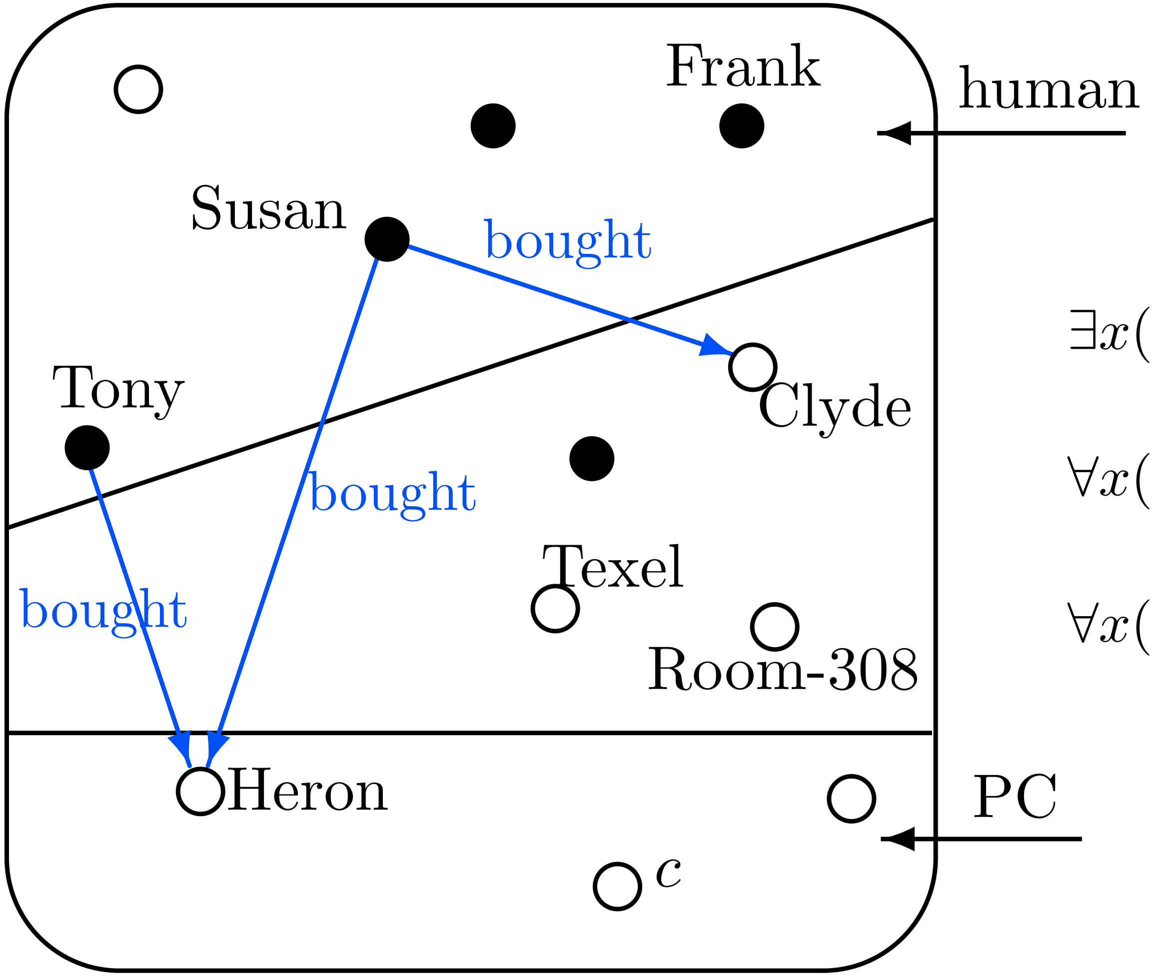
Hence,

$$M \models \forall x(\texttt{bought}(\texttt{Tony}, x) \to \texttt{bought}(\texttt{Susan}, x)).$$

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 ●s are the lecturers



$\exists x(\texttt{PC}(x) \wedge \texttt{bought}(\texttt{Frank}, x))$

$\forall x(\texttt{lecturer}(x) \rightarrow \texttt{human}(x))$

$\forall x(\exists y\ \texttt{bought}(y, x) \rightarrow \neg\texttt{human}(x))$

Rough guide: advice

For a fairly complex formula like $\exists x(\texttt{PC}(x) \wedge \exists y\, \texttt{bought}(y, x))$:

Work out what each subformula says in English, working from atomic subformulas (leaves of formation tree) up to the whole formula (root of formation tree).

- $\exists x(\texttt{PC}(x) \wedge \exists y\, \texttt{bought}(y, x))$
  - $\texttt{PC}(x) \wedge \exists y\, \texttt{bought}(y, x)$
    - $\texttt{PC}(x)$
    - $\exists y\, \texttt{bought}(y, x)$
      - $\texttt{bought}(y, x)$

- something bought a PC
  - x is a PC & something bought x
    - x is a PC
    - something bought x
      - y bought x

This is often a good guide to evaluating the formula.
E.g., the formula here says that there is an x that's a PC and that something bought it, (it's pointed to by an arrow). So look for one.

Alessandra Russo
C140 Logic - 1st year course

Truth in a structure — formally!

We saw how to evaluate some formulas in a structure 'by inspection'.

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

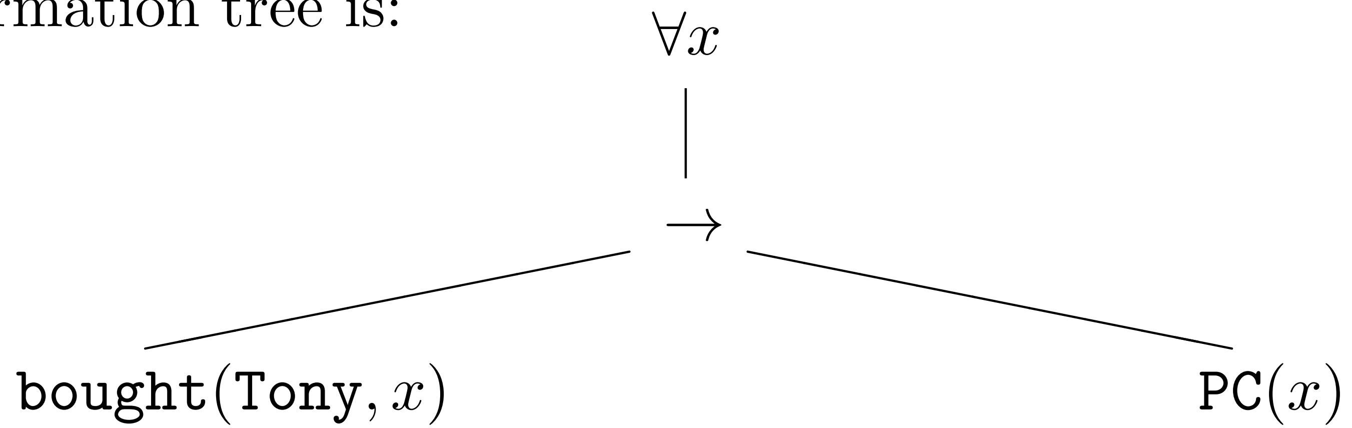
We need a more formal way to evaluate all predicate logic formulas in structures.

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, thing are not so simple. . .

A problem

$\forall x(\texttt{bought}(\texttt{Tony}, x) \to \texttt{PC}(x))$ is true in the structure M on slide 34.
Its formation tree is:



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?!

Not all formulas of predicate logic are true or false in a structure!

What's going on?