Classical First-Order Predicate Logic

o## Why Predicate logic?

It is a powerful extension of propositional logic

Predicate logic in a nutshell

Predicate logic is concerned with describing **relationships between objects**, and the ways in which different relationships are logically connected.

So, atomic formulas become **structured**.

Syntactically, there are 6 new features:

- Predicates (that take arguments): $\mathrm{sister(Bob, Mary), \ldots.}$
- Constants: $\mathrm{Bob, Room\_308, \ldots.}$
- Variables: $\mathrm{x, y, z, \ldots.}$
- Quantifiers: ∀ (for all), ∃ (there exists)
- Functions: $\mathrm{father\_of(\_), sum\_of(\_,\_), +, -, x, \ldots}$
- Equality: $=$

Semantically, the notion of **situation** in predicate logic is more complex than in propositional logic. We have to give meaning to constants, functions, predicates and variables.

Syntax — The formal language

First, we need to fix a precise definition of the formal language, that is the **syntax** of predicate logic.

Splitting the atoms - new atomic formulas

In propositional logic, we regard phrases like '*Arron is a student*' and '*Arron and Russell are friends*' as atomic, without an internal structure. Now we look inside!

We regard “*being a student*” as a **property** that Arron (and other objects) may, or may not, have; “*being friends*” as a relation that Arron and Russell (and other two objects) may, or may not, have

So we introduce:

- **Predicate symbols**, to describe properties of and relations between objects
    - student: Takes 1 argument – it is **unary**, or its 'arity' is 1
    - friends: Takes 2 arguments – it is **binary**, or its 'arity' is 2
- **Constants**, to name objects
    - $\text{Arron}, \text{Russell}, \ldots,$

Then $\text{student}(\text{Arron})$ and $\text{friends}(\text{Arron}, \text{Russell})$ are examples of new **atomic formulas**

Quantifiers

Predicate logic uses **quantifiers** to vary the arguments to friends

This allows us to express characteristics about the relationship 'friends'

What are quantifiers?

A quantifier specifies a quantity (of objects that have some property)

Example: '*All students work hard*'

Quantifiers in predicate logic

There are just two:

- ∀: ‘for all’
- ∃: ‘there exists’ (or ‘some’)

Some other quantifiers can be expressed with these. They can also express each other.

But quantifiers like *infinitely many* and *more than* cannot be expressed in first-order logic in general.

Expressions like $\mathrm{Russell}$, Arron, etc are constants. So, to express '*Arron is a friend of everyone*' we need **variables** that range over all people, not just $\mathrm{Russell}$, etc.

Variables

We use **variables** to do quantification

We fix an infinite set V of variables: e.g., $x, y, z, u, v, w, x_0, x_1, x_2, \ldots$

Now you can write formulas like $\mathrm{student}(x)$

- To say '*Everyone is a student*', we'll write $\forall x\ \mathrm{student}(x)$
- '*Someone is a student*', can be written as $\exists x\ \mathrm{student}(x)$
- '*Frank has a student friend*', can be written $\exists x\ (\mathrm{student}(x) \land \mathrm{friend}(\mathrm{frank}, x))$

Function symbols

In arithmetic (and Haskell) we are used to **functions**, such as $+$, $-$, $\times$, etc

Predicate logic can do this too

- A function symbol refers to an object in terms of another object or objects

A function symbol comes with a fixed **arity** (number of arguments)

Examples of functions: $\mathrm{father\_of}(\_), \mathrm{mother\_of}(\_), \mathrm{sum\_of}(\_, \_)$

Equality

$t_1 = t_2$ means "t_1 and t_2 refer to the same object"

Signatures

Definition 4.1 (signature)

The *signature* of a predicate logic is a triple $\langle \mathcal{K}, \mathcal{F}, \mathcal{P} \rangle$, where $\mathcal{K}$ is a (possibly empty) set of constants, $\mathcal{F}$ is a (possibly empty) set of function symbols, and $\mathcal{P}$ is a set of predicate symbols. Function and predicate symbols have specific arities.

It replaces the collection of propositional atoms you have seen in propositional logic.

We usually write L to denote a signature. We often write $\mathrm{c}, \mathrm{d}, \ldots$ for constants, $\mathrm{f}, \mathrm{g}, \ldots$ for function symbols and $\mathrm{P}, \mathrm{Q}, \mathrm{R}, \mathrm{S}, \ldots$ for predicate symbols

An Example

Which symbols we put in L depends on what we want to say.

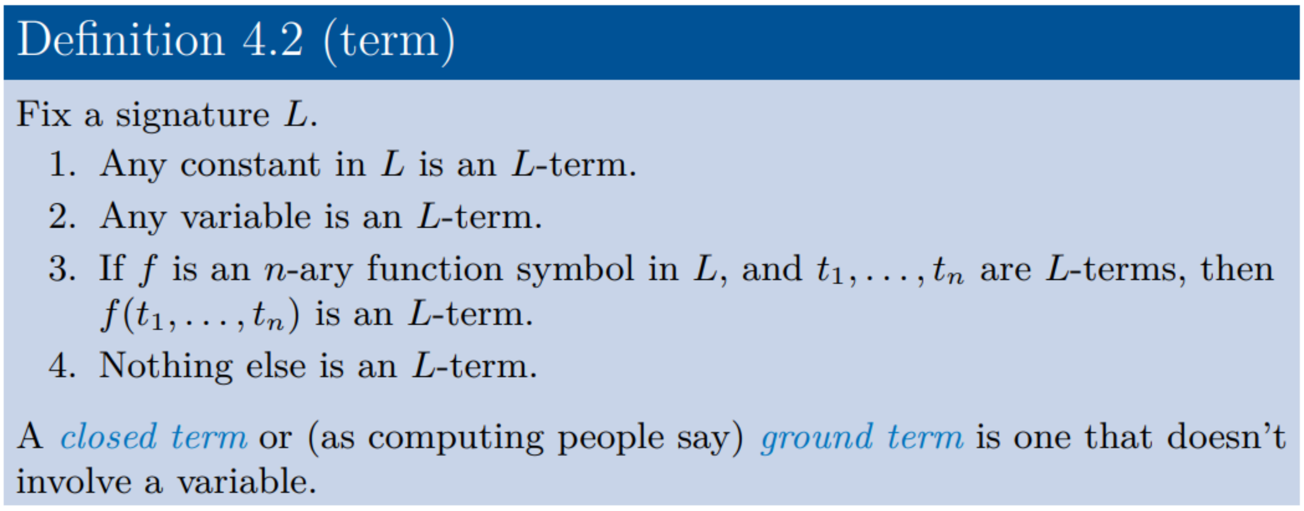
For illustration, we'll use a signature L consisting of:

- Constants frank, susan, tony, heron, clyde, and c
- Unary function symbol $\mathrm{father\_of}$ (arity 1) (also $\mathrm{father\_of}/1$)
- Unary predicate symbols PC, Human, $\mathrm{Lecturer}$ (arity 1) (also $\mathrm{PC}/1$, $\mathrm{Human}/1$, $\mathrm{Lecturer}/1$)
- A binary predicate symbol Bought(arity 2) (also $\mathrm{Bought}/2$)

Warning: things in L are just symbols — syntax. They don't come with any meaning. To give them meaning, we'll need to work out (later) what a **situation** in predicate logic should be

Terms

To write formulas, we need **terms** to name objects. Terms are not formulas. They will not be true or false.



Definition 4.2 (term)

Fix a signature L.

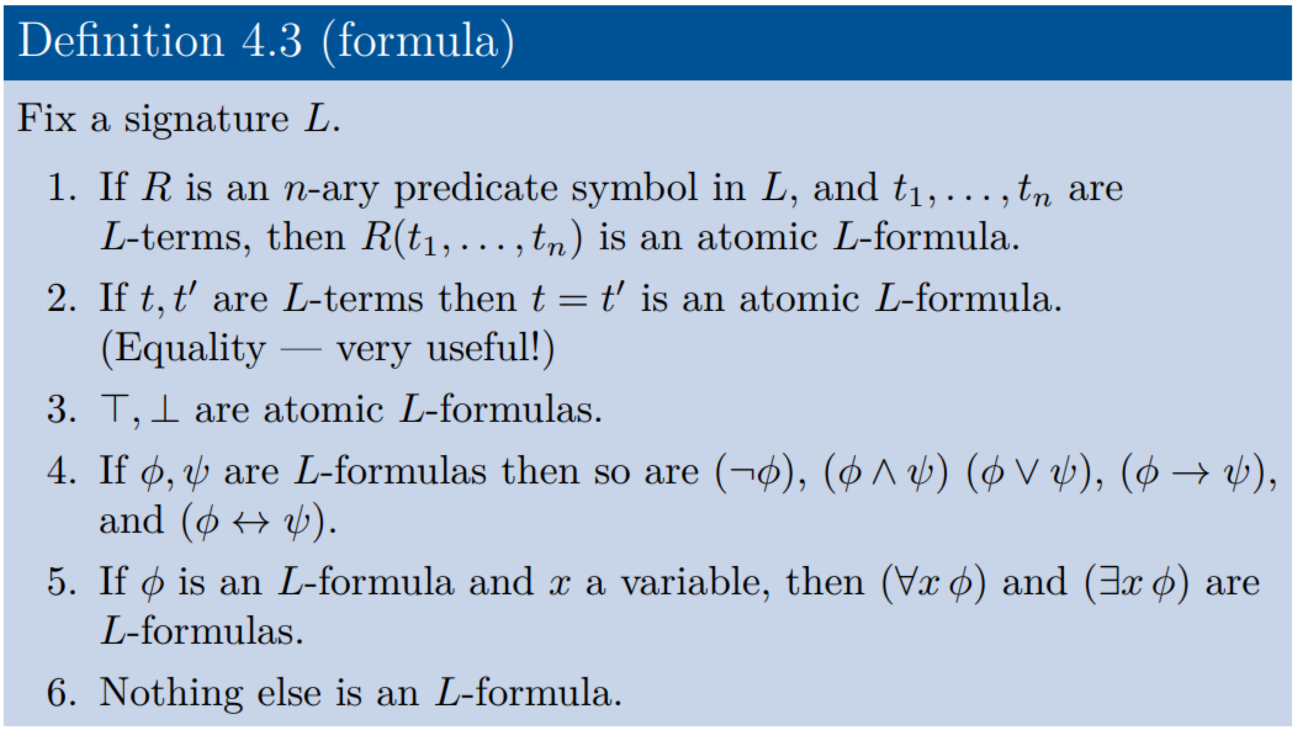
1. Any constant in L is an L-term.
2. Any variable is an L-term.
3. If f is an n-ary function symbol in L, and $t_1, \ldots, t_n$ are L-terms, then $f(t_1, \ldots, t_n)$ is an L-term.
4. Nothing else is an L-term.

A *closed term* or (as computing people say) *ground term* is one that doesn't involve a variable.

Examples:

- $frank$, $father\_of(susan)$: ground terms
- x, $g(x)$, $father\_of(g(y))$: not ground terms

Formulas of first-order logic



Definition 4.3 (formula)

Fix a signature L.

1. If R is an n-ary predicate symbol in L, and $t_1, \ldots, t_n$ are L-terms, then $R(t_1, \ldots, t_n)$ is an atomic L-formula.
2. If t, t' are L-terms then $t = t'$ is an atomic L-formula. (Equality — very useful!)
3. $\top, \bot$ are atomic L-formulas.
4. If ϕ, ψ are L-formulas then so are $(\neg\phi)$, $(\phi \wedge \psi)$ $(\phi \vee \psi)$, $(\phi \to \psi)$, and $(\phi \leftrightarrow \psi)$.
5. If ϕ is an L-formula and x a variable, then $(\forall x\, \phi)$ and $(\exists x\, \phi)$ are L-formulas.
6. Nothing else is an L-formula.

Atomic formulas, Literals, and Sentences

An **atomic formula** is a predicate symbol with arguments filled in with terms:

- $\mathrm{Lecturer}(\mathrm{susan})$
- $\mathrm{PC}(x)$

A **literal** is an atomic formula or its negation:

- $\mathrm{Sum\_of}(x, 4) = 10$
- $\neg\mathrm{Lecturer}(\mathrm{mother\_of}(x))$

A **sentence** is a formula with all variables in the **scope** of a quantifier

- $\forall x \forall y\, \mathrm{Bought}(x, y)$

Binding conventions: as for propositional logic, plus: $\forall x$, $\exists x$ have same strength as $\neg$

Formation trees, sub-formulas and clauses can be done much as before

Examples of Formulas

1. $\mathtt{Bought}(\mathtt{frank}, x)$

2. $\neg\mathtt{Bought}(\mathtt{frank}, x)$

3. $\exists x\, \mathtt{Bought}(\mathtt{frank}, x)$

4. $\forall x(\mathtt{Lecturer}(x) \to \mathtt{Human}(x))$

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

6. $\forall x(\texttt{Bought}(\texttt{father\_of}(\texttt{tony}), x) \to \texttt{Bought}(\texttt{susan}, x))$
   'Susan bought everything that Tony's father bought.'
7. $\forall x\,\texttt{Bought}(\texttt{father\_of}(\texttt{tony}), x) \to \forall x\,\texttt{Bought}(\texttt{susan}, x)$
   'If Tony's father bought everything, so did Susan.'
   Note the difference!
8. $\forall x \exists y\,\texttt{Bought}(x, y)$
   'Everything bought something.'
9. $\exists y \forall x\,\texttt{Bought}(x, y)$
   'There is something that everything bought.'
   Note the difference!
10. $\exists x \forall y\,\texttt{Bought}(x, y)$
    'Something bought everything.'

Predicate logic is powerful and terse

Semantics—The meaning

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 need first to give meaning to all the symbols in a given signature

We then need to define the notions of valuation of terms and evaluation for atomic formulas, and quantifiers

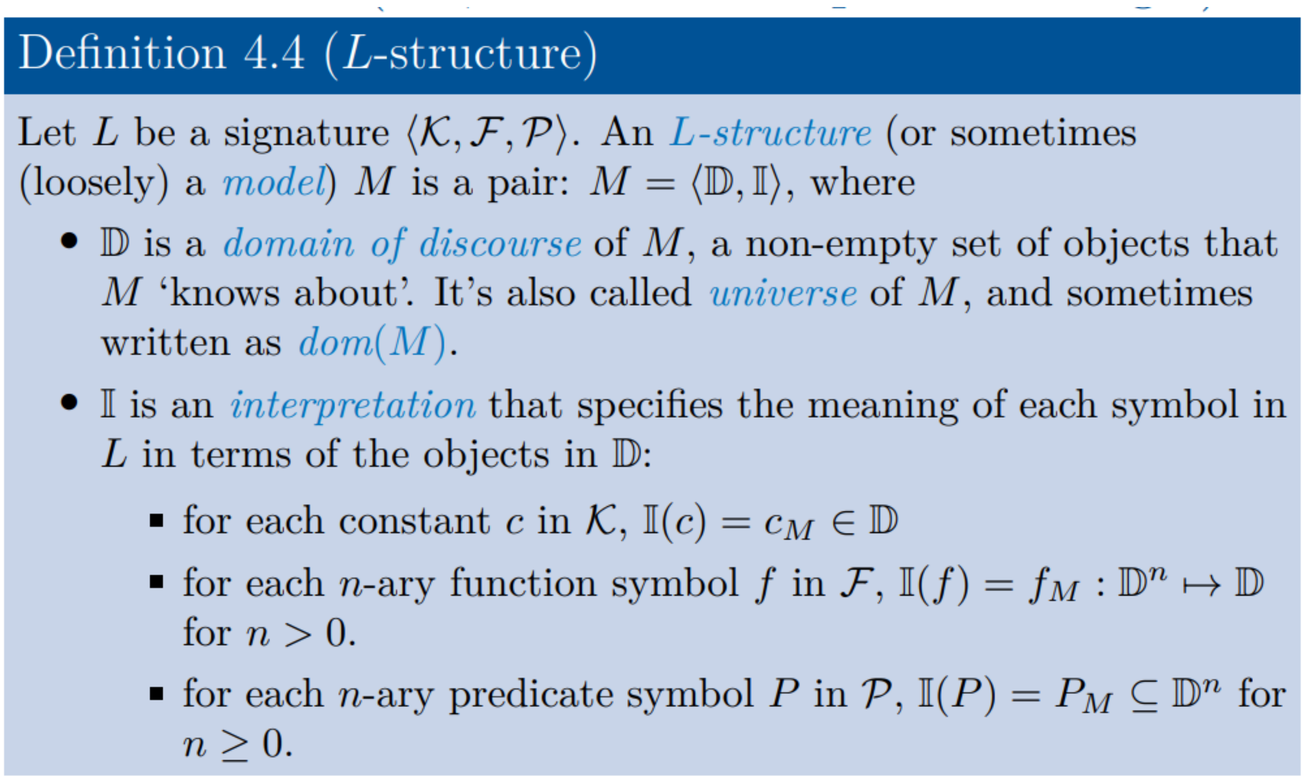
Functions and relations over a Domain of Discourse

A collection of objects to which a predicate logic might refer is called a **domain of discourse**

- A **domain of discourse** is a **non-empty** set of objects. We denote it with D

- For a positive whole number n, the n-ary **Cartesian power** of a set D, written D^n or $D\times,...,\times D$, is the set of all n-tuples that can be constructed from its members
- A **relation** R of arity n over D is a subset of D^n, so $R \subseteq D^n$, with $n \geq 0$
- A **unary relation** R is a relation of arity 1, so $R \subseteq D$
- A **binary relation** R is a relation of arity 2. So $R \subseteq D^2$
- A **function of arity n** over D is a one-to-one or a many-to-one mapping from all of D^n to D, with $n > 0$
- A **many-to-one function** means that different n-tuples of objects in D may be mapped to the same object in D, but each n-tuple is mapped to exactly one object

Structures (i.e., situations in predicate logic)



Definition 4.4 (L-structure)

Let L be a signature $\langle \mathcal{K}, \mathcal{F}, \mathcal{P} \rangle$. An *$L$-structure* (or sometimes (loosely) a *model*) M is a pair: $M = \langle \mathbb{D}, \mathbb{I} \rangle$, where

- $\mathbb{D}$ is a *domain of discourse* of M, a non-empty set of objects that M 'knows about'. It's also called *universe* of M, and sometimes written as $dom(M)$.
- $\mathbb{I}$ is an *interpretation* that specifies the meaning of each symbol in L in terms of the objects in $\mathbb{D}$:
  - for each constant c in $\mathcal{K}$, $\mathbb{I}(c) = c_M \in \mathbb{D}$
  - for each n-ary function symbol f in $\mathcal{F}$, $\mathbb{I}(f) = f_M : \mathbb{D}^n \mapsto \mathbb{D}$ for $n > 0$.
  - for each n-ary predicate symbol P in $\mathcal{P}$, $\mathbb{I}(P) = P_M \subseteq \mathbb{D}^n$ for $n \geq 0$.

0-ary predicates are similar to propositional atoms in Propositional Logic. The 0-ary Cartesian power of D, denoted as D^0, is defined to be a singleton set containing the empty set, i.e., $D^0 = \{\emptyset\}$

Example of a structure $M = \langle D, I \rangle$

For our simple signature L, an L-structure should define the domain and the meaning of the symbols in L

Let D be a set of 12 objects: $\mathrm{D} = \{o_1, o_2, \ldots, o_{12}\}$

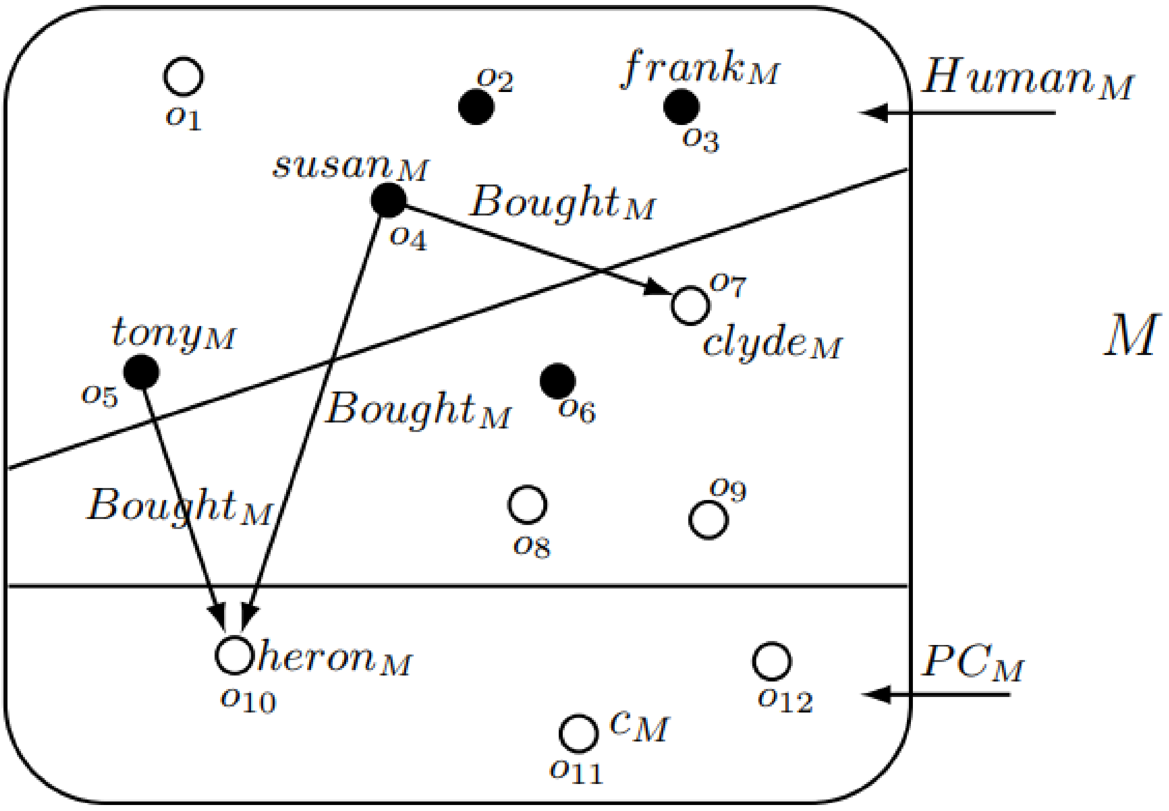
Let I be the following interpretation:

- $\mathbb{I}(\texttt{tony}) = o_5$, $\mathbb{I}(\texttt{susan}) = o_4$, $\mathbb{I}(\texttt{frank}) = o_3$, $\mathbb{I}(\texttt{clyde}) = o_7$, $\mathbb{I}(c) = o_{11}$, $\mathbb{I}(\texttt{heron}) = o_{10}$.
- $\mathbb{I}(father\_of) = f_M : \mathbb{D} \mapsto \mathbb{D}$, given by: $\langle o_1 \mapsto o_1, o_2 \mapsto o_1, o_3 \mapsto o_1, o_4 \mapsto o_2, o_5 \mapsto o_2, o_6 \mapsto o_7, o_7 \mapsto o_8, o_8 \mapsto o_1, o_9 \mapsto o_6, o_{10} \mapsto o_{11}, o_{11} \mapsto o_{12}, o_{12} \mapsto o_{12} \rangle$.
- $\mathbb{I}(\texttt{Human}) = \{o_1, o_2, o_3, o_4, o_5\}$, $\mathbb{I}(\texttt{Lecturer}) = \{o_2, o_3, o_4, o_5, o_6\}$, $\mathbb{I}(PC) = \{o_{10}, o_{11}, o_{12}\}$.
- $\mathbb{I}(\texttt{Bought}) = \{(o_4, o_{10}), (o_5, o_{10}), (o_4, o_7)\}$

Depicting the structure $\mathrm{M} = \langle \mathrm{D}, \mathrm{I} \rangle$ as a diagram

When we have just unary and binary predicates, we can depict a structure as a diagram.

This is a diagram of our L-structure M



- The 12 dots are the 12 objects in $\mathrm{dom}(M)$
- Some objects have an additional label (e.g. frank_M) to indicate the meaning of the constants in L
- 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
- We always omit in the diagram the interpretation of the function symbols to avoid confusion with that of the predicate symbols

Drawing other symbols

Our simple signature L has constants, one unary function symbol, unary and binary predicate 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 predicate symbols

- Drawing arrows between the objects that M says satisfy the binary predicate symbols. The arrow direction matters

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

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

We do not draw the interpretation of function symbols, but specify it mathematically

0-ary predicate symbols are the same as propositional atoms

Truth in a structure (informally)

When is a formula **without quantifiers and variables** true in a structure?

$PC(heron)$ is true in M, because the interpretation of the constant $heron$ is an object (o_{10}) that is in the interpretation of PC, that is $I(heron) \in I(PC)$ or in other terms $o_{10} \in \{o_{10}, o_{11}, o_{12}\}$

We write this as $M \models PC(heron)$

Read as '*M says* $PC(heron)$'

Warning: This is a different use of $\models$ from what seen in the definition of valid argument

'$\models$' is overloaded — it's used for two different things

$\mathrm{Bought}(susan, susan)$ is false in M, because M does not say that the object labelled $susan_M(o_4)$ "*bought*" itself, i.e.
$(I(susan), I(susan)) \notin I(\mathrm{Bought})$, i.e.,
$(o_4, o_4) \notin \{(o_4, o_{10}), (o_5, o_{10}), (o_4, o_7)\}$

We write this as $M \not\models \mathrm{Bought}(susan, susan)$

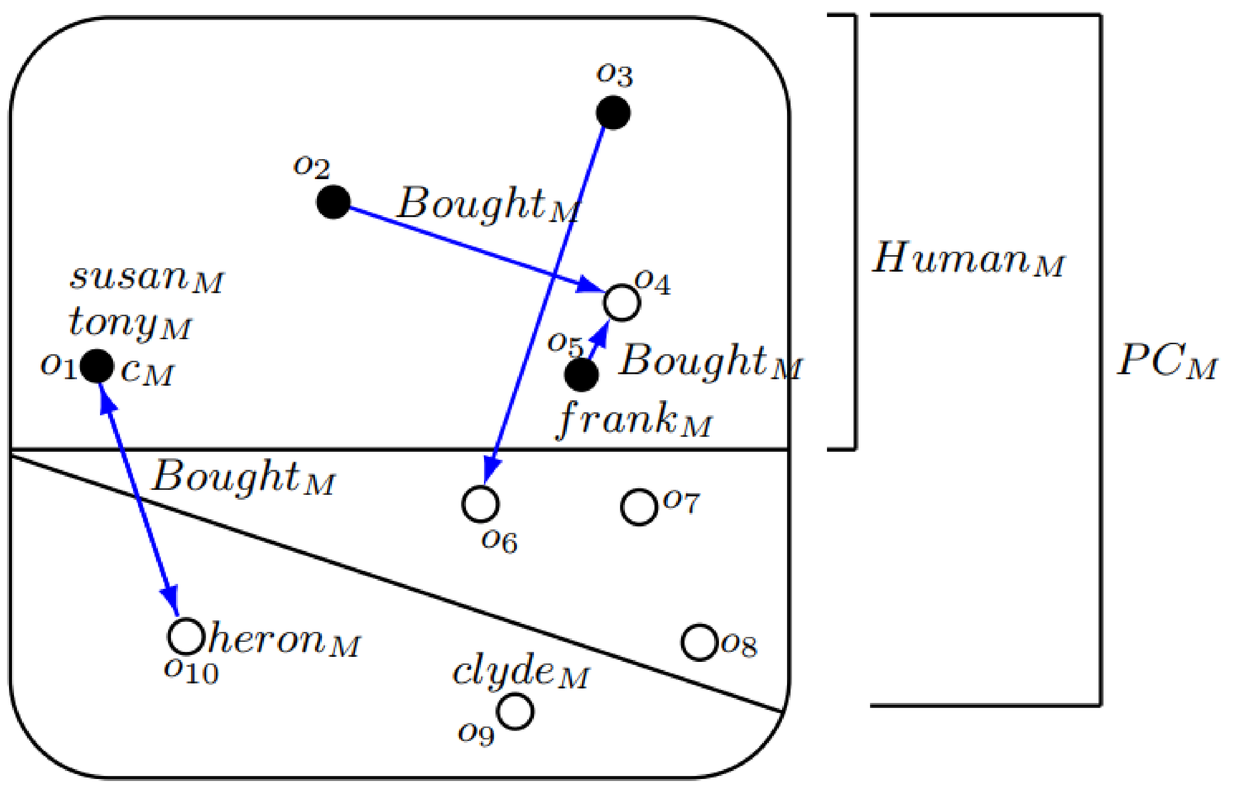
From our knowledge of propositional logic:

$M \not\models PC(tony) \vee \mathrm{Bought}(father\_of(frank), clyde)$

because $o_5 \notin \{o_{10}, o_{11}, o_{12}\}$ and $(o_1, o_7) \notin \{(o_4, o_{10}), (o_5, o_{10}), (o_4, o_7)\}$

Another structure

Here is another L-structure, called M'. $\mathbb{D} = \{o_1, \ldots, o_{10}\}$ and the interpretation $\mathbb{I}(father\_of) = \langle o_1 \mapsto o_2, o_2 \mapsto o_3, o_3 \mapsto o_6,$ $o_4 \mapsto o_5, o_5 \mapsto o_7, o_6 \mapsto o_7, o_7 \mapsto o_8, o_8 \mapsto o_9, o_9 \mapsto o_{10}, o_{10} \mapsto o_{10} \rangle$.



Some statements about M'

$M' \not\models \text{Bought}(father\_of(\text{susan}), \text{clyde})$

$M' \models \texttt{susan} = \texttt{tony}?$

$M' \models \texttt{Human}(\texttt{tony}) \wedge PC(\texttt{frank})?$

$M' \models \texttt{Bought}(\texttt{tony}, father\_of(\texttt{heron})) \wedge \texttt{Bought}(Heron, c)?$

$$M' \models \texttt{Bought}(\texttt{susan}, \texttt{clyde}) \rightarrow \texttt{Human}(\texttt{clyde})\ ?$$