First-Order Structures

Definition: Given a first order language (over) $\mathcal{L}$, (with $\mathcal{L} = \mathcal{A} \cup \mathcal{F} \cup \mathcal{P}$), an *$\mathcal{L}$-structure* is a pair $\mathbf{M} = (M, I)$ where M is a non-empty set called the *domain* (or *carrier*) of the structure, and I is a function called the *interpretation function* which assigns meaning (in M) to the symbols in $\mathcal{L}$ as follows:

1. For every constant symbol $a \in \mathcal{A}$, $I(a)$ is some element of M.
2. For every function symbol $f \in \mathcal{F}$ with $arity_{\mathcal{F}}(f) = n$, $I(f) : M^n \to M$.
3. For every predicate symbol $p \in \mathcal{P}$ with $arity_{\mathcal{P}}(p) = n$, $I(p) : M^n \to \{true, false\}$. Note that for predicate symbols of arity 0, this reduces to a valuation.

We will use the following notation which avoids reference to the function I: $a_{\mathbf{M}} = I(a)$, $f_{\mathbf{M}} = I(f)$, and $p_{\mathbf{M}} = I(p)$.

Assignments

Definition: Given a first order language (over) $\mathcal{L}$, and an $\mathcal{L}$-structure, $\mathbf{M}$, an *assignment* is any function $s : \mathcal{X} \to M$. (The set of all such functions is denoted by $[\mathcal{X} \to M]$.)

Definition: Given a domain M and an element $a \in M$, and an assignment s, the assignment $s[x_i := a]$ denotes a new assignment s' such that:

$$s'(y) = s(y) \text{ for all } y \neq x_i, \text{ and } s'(x_i) = a.$$

Assigning Meaning to Terms

Definition: Given a first order language (over) $\mathcal{L}$, and an $\mathcal{L}$-structure, $\mathbf{M}$, The interpretation of a term t in $\mathbf{M}$, $I(t)$, is as a function which, given an assignment, returns a value in the domain. That is, $t_{\mathbf{M}} : [\mathcal{X} \to M] \to M$ defined by a term t is a function such that, given an assignment $s \in [\mathcal{X} \to M]$ returns the value of the term under that assignment. That value, $t_{\mathbf{M}}(s)$, is defined recursively as follows:

1. If $t = a \in \mathcal{A}$,then $t_{\mathbf{M}}(s) = a_{\mathbf{M}}$.
2. If $t = x \in \mathcal{X}$,then $t_{\mathbf{M}}(s) = s(x)$.
3. If $t = f(t_1, \ldots, t_n)$ and $f \in \mathcal{F}$ with $arity_{\mathcal{F}}(f) = n$, and $t_1, \ldots, t_n \in TERM$, then $t_{\mathbf{M}}(s) = f_{\mathbf{M}}((t_1)_{\mathbf{M}}(s), \ldots, (t_n)_{\mathbf{M}}(s))$.

Assigning Meaning to Atomic Formulas

Definition: Given a first order language (over) $\mathcal{L}$, an $\mathcal{L}$-structure, $\mathbf{M}$, and A, an atomic formula of $\mathcal{L}$, $I(A)$ is a function which, given an assignment, returns a truth value for the term. That is, $A_{\mathbf{M}} : [\mathcal{X} \to M] \to \{true, false\}$ defined by an atomic formula A is a function such that, for every assignment $s \in [\mathcal{X} \to M]$ the value of $A_{\mathbf{M}}(s)$ is defined recursively as follows:

- Atomic Formulas:
  1. If $A = \bot$, then $A_{\mathbf{M}}(s) = false$.
  2. If $A = \top$, then $A_{\mathbf{M}}(s) = true$.
  3. If $A = p(t_1, \ldots, t_n)$ and $p \in \mathcal{P}$ with $arity_{\mathcal{P}}(p) = n$, and $t_1, \ldots, t_n \in TERM$, then $A_{\mathbf{M}}(s) = p_{\mathbf{M}}((t_1)_{\mathbf{M}}(s), \ldots, (t_n)_{\mathbf{M}}(s))$.

Assigning Meaning to Non-Atomic Formulas

- Compound Formulas:

  1. If $A = (\neg A_1)$, then $A_{\mathbf{M}}(s) = \neg_{\mathbf{M}}(A_{1\mathbf{M}}(s))$.
  2. If $A = (A_1 \bullet A_2)$, with $\bullet \in \{\wedge, \vee, \Rightarrow, \equiv\}$, then $A_{\mathbf{M}}(s) = (A_{1\mathbf{M}}(s) \bullet_{\mathbf{M}} A_{2\mathbf{M}}(s))$.

  where $\neg_{\mathbf{M}}$ and $\bullet_{\mathbf{M}}$ are the truth-functionals for these operators, defined as before, but modified to accept assignments.

- Quantified Formulas:

  1. If $A = (\forall x(A_1))$, then $A_{\mathbf{M}}(s) = true$ iff for **all** $m \in M$, $A_{1\mathbf{M}}(s[x := m]) = true$.
  2. If $A = (\exists x(A_1))$, then $A_{\mathbf{M}}(s) = true$ iff for **some** $m \in M$, $A_{1\mathbf{M}}(s[x := m]) = true$.

Satisfiability

Definition: let $\mathcal{L}$ be a first-order language .

- Given a formula A, an $\mathcal{L}$-structure $\mathbf{M}$, and an assignment s, we say that $\mathbf{M}$ *satisfies* A *with (*or*, under)* s, written:

  $$\mathbf{M} \models A(s)$$

  iff $A_{\mathbf{M}}(s) = true$.
- Given a formula A, and an $\mathcal{L}$-structure $\mathbf{M}$, we say that $\mathbf{M}$ *satisfies* A, or, A *is satisfiable in* $\mathbf{M}$, iff there is some assignment s such that $A_{\mathbf{M}}(s) = true$.
- Given a formula A, we say that A *is satisfiable* iff there is an $\mathcal{L}$-structure $\mathbf{M}$ and some assignment s such that $A_{\mathbf{M}}(s) = true$.

Validity

Definition: let $\mathcal{L}$ be a first-order language .

- Given a formula A and an $\mathcal{L}$-structure $\mathbf{M}$, we say that A *is valid in* $\mathbf{M}$, written:

  $$\mathbf{M} \models A$$

  iff $A_{\mathbf{M}}(s) = \mathit{true}$ for all assignments s. We also say, in such a case, that $\mathbf{M}$ *is a model of* A, or that $\mathbf{M}$ *models of* A.

- Given a formula A, we say that A *is valid*, written:

  $$\models A$$

  iff it is valid in every $\mathcal{L}$-structure (i.e. every $\mathcal{L}$-structure models A).

Satisfiability, Validity of a Set

These definitions extend to a set of formulas iff they apply to all the formulas in the set using a single structure and/or assignment. I.e.:

Definition: A set of formulas, Γ, is mutually satisfiable iff there is an $\mathcal{L}$-structure $\mathbf{M}$ and an assignment s such that $\mathbf{M} \models A_i(s)$ for all $A_i \in \Gamma$.

As in the propositional case, this is different from saying that a set is satisfiable if each element is satisfiable. It is even different from saying that each is satisfiable by a given model $\mathbf{M}$.

Definition: A set of formulas, Γ, is valid in an $\mathcal{L}$-structure $\mathbf{M}$ (that is, $\mathbf{M}$ *is a model of* Γ) if $\mathbf{M} \models A_i$ for all $A_i \in \Gamma$.

Consequence

Definition: Given a set of formulas Γ and a formula A, we say that A *is a semantic consequence of* Γ (or, A *is a logical consequence of* Γ), written:

$$\Gamma \models A$$

iff for every $\mathcal{L}$-structure $\mathbf{M}$, and every assignment s, if $\mathbf{M} \models A_i(s)$ for all $A_i \in \Gamma$, then $\mathbf{M} \models A$.

Notes on Validity and Consequence

It is important to get a sense early of how strong a notion first-order validity is, particularly as it applies to the universal quantifier.

First of all, it means that unlike propositional logic, there is no semantic decision procedure to determine validity of a first-order formula.

In propositional logic it was possible to enumerate all the possible meanings of a formula by building a truth table from all the assignments of truth values to the propositional letters. Though this is exponential, it is still decidable.

In first-order we would need to enumerate all possible assignments of values to the constants and variables. Since domains may be infinite, and since there are certainly an infinite number of domains, there is no way to approach this problem.

Notes on Validity and Consequence

Certainly there are lots of valid formulas. For instance, $\forall x(p(x) \Rightarrow p(x))$ is valid.

In fact, if we take any propositional tautology and parameterize the propositional letters by a variable and then universally quantify that variable at the outside of the formula, then the resulting formula is valid in first-order logic.

Another example of a valid formula is

$$(\forall x(p(x))) \Rightarrow (\exists x(p(x)))$$

Note that this formula is only valid because we assumed domains were non-empty.

In contrast, the formula:

$$(\exists x(p(x))) \Rightarrow (\forall x(p(x)))$$

is valid in some models (for example, in any one element domain), but not all.

Notes on Validity and Consequence

One must be sensitive to the implication of the requirement of being true in *all* models.

Suppose we try to give a first-order definition of some sets, and the notion of subset containment as follows:

- $t_1 = \{a, b\}$:
$$in(t_1, a) \wedge in(t_1, b)$$
- $t_2 = \{a, b, c\}$:
$$in(t_2, a) \wedge in(t_2, b) \wedge in(t_2, c)$$
- One set is a subset of another iff when something is an element of the first, it is also an element of the second:

  $\forall s_1(\forall s_2(subset(s_1, s_2) \equiv (\forall x(in(s_1, x) \Rightarrow in(s_2, x)))))$

If we take Γ to be those three formulas, does

$$\Gamma \models subset(t_1, t_2)$$

hold?

Notes on Validity and Consequence

While it might appear that it should, it doesn't.

Certainly any $\mathcal{L}$-structure in which we map t_1 and its elements to a two element set and t_2 an its elements to a three element superset of that set, and map in and $subset$ to $\in$ and $\subset$, respectively, models Γ, and also $subset(t_1, t_2)$.

But suppose we map to the domain of the natural numbers, and map a to 2, b to 4, and c to 7 and in and $subset$ to $\in$ and $\subset$, respectively. Suppose, further, that t_1 is mapped to the even numbers, and t_2 is mapped to the numbers less than 10.

This $\mathcal{L}$-structure models all three formulas in Γ, but t_1 is not a subset of t_2 in this model!

The problem is that while we specify the things that must be true, we cannot assume that this is all that is true. Just because we say that a and b are in t_1 does not mean there might not be other members of that set as well.

Notes on Validity and Consequence

But to prove a universal statement it must be true of all elements (including the ones we haven't mentioned in our assumptions) in all models.

The quantifiers are called *intensional* (as opposed to *extensional*) and it means that we cannot express our intent via enumeration, but only by other universal statements.

We can limit ourselves to models where t_1 is a subset of t_2 (and thereby make the consequence valid) by adding any of the following assumptions to Γ:

- t_1 is a subset of t_2:
$$subset(t_1, t_2)$$
- Any element of t_1 is also an element of t_2:
$$\forall x(in(t_1, x) \Rightarrow in(t_2, x))$$
- Any other object we haven't mentioned is neither in t_1 nor in t_2:
$$\forall x((\neg(x = a) \wedge \neg(x = b)) \Rightarrow \neg in(t_1, x))$$
$$\forall x((\neg(x = a) \wedge \neg(x = b) \wedge \neg(x = c)) \Rightarrow \neg in(t_2, x))$$

The Closed World Assumption

The last pair of assumptions is similar in intent to the *closed world assumption*, which says that we restrict our domain to worlds that have only enough objects to account for the ones we have explicitely mentioned in our assumptions and consequence formulas, and that every predicate not stated to be true is in fact false.

Under the closed-world assumption, intensional quantification and *extensional quantification* (in which enumeration is sufficient) are the same.

The closed world assumption was implicit in our discussion of database reasoning. We assumed that the database was a complete portrayal of the domain of discourse. If the database doesn't say Josh drinks Budweiser, then he doesn't.

The closed world assumption is also used in explaining the use of negation in Prolog.