Last Week

Applying Propositional Logic

- Computability and Decidability
- Boolean Circuits
- Boolean Satisfiability (SAT)

Outline

- First-Order Logic: Motivation
- First-Order Logic: Syntax
- First-Order Logic: Semantics
- Definability

First-Order Logic: Motivation

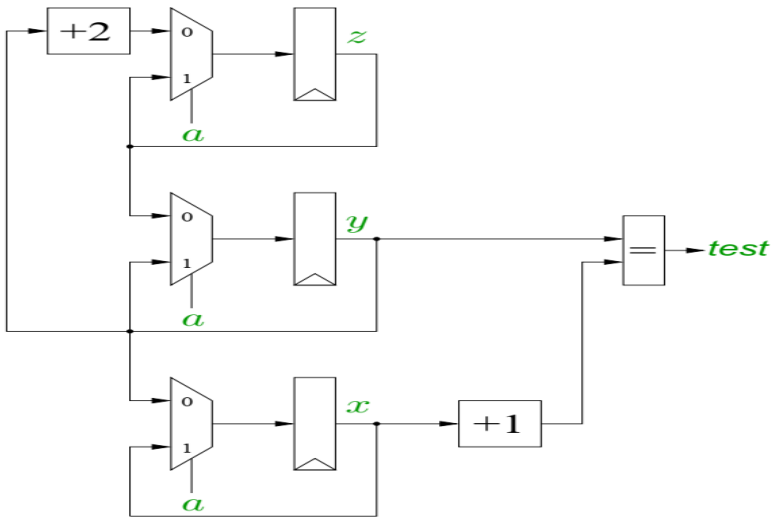
Propositional logic is not powerful enough for many applications.

For example, propositional logic cannot reason about natural numbers directly.

In general, to reason about infinite domains or to express properties which are more abstract, a more expressive logic is required.

First-order logic is the most common logic of choice for handling tasks that require more power than that offered by propositional logic.

Circuit Example



Circuit Example

One way to prove the property of the circuit is by induction.

The inductive step is essentially the following:

```
(y = x + 1 AND z = x + 2 AND
 x' = IF a THEN x ELSE y AND
 y' = IF a THEN y ELSE z AND
 z' = IF a THEN z ELSE y + 2) IMPLIES
 y' = x' + 1 AND z' = x' + 2
```

We can prove this formula by showing that the negation is unsatisfiable.

We can write this formula in propositional logic by using one propositional variable for each bit in the current and next states.

Assuming a bit-width of 2 for simplicity and skipping the details, we get the following formula:

$$
\begin{array}{l}
(z1 \leftrightarrow \neg x1) \wedge (z0 \leftrightarrow x0) \wedge \\
(y1 \leftrightarrow (x1 \oplus x0)) \wedge (y0 \leftrightarrow \neg x0) \wedge \\
(a \rightarrow ((xp1 \leftrightarrow x1) \wedge (xp0 \leftrightarrow x0))) \wedge \\
(\neg a \rightarrow ((xp1 \leftrightarrow y1) \wedge (xp0 \leftrightarrow y0))) \wedge \\
(a \rightarrow ((yp1 \leftrightarrow y1) \wedge (yp0 \leftrightarrow y0))) \wedge \\
(\neg a \rightarrow ((yp1 \leftrightarrow z1) \wedge (yp0 \leftrightarrow z0))) \wedge \\
(a \rightarrow ((zp1 \leftrightarrow z1) \wedge (zp0 \leftrightarrow z0))) \wedge \\
(\neg a \rightarrow ((zp1 \leftrightarrow \neg y1) \wedge (zp0 \leftrightarrow y0))) \wedge \\
(\neg(zp1 \leftrightarrow \neg xp1) \vee \neg(zp0 \leftrightarrow xp0) \vee \\
\neg(yp1 \leftrightarrow (xp1 \oplus xp0)) \vee (yp0 \leftrightarrow xp0))
\end{array}
$$

Note that the formula for an n-bit counter requires a propositional formula of size $O(n)$.

Using first-order logic, we can use a formula whose size is constant for all n:

$$
\begin{aligned}
(\quad & y = x +_{[2^n]} 1 \wedge z = x +_{[2^n]} 2 \wedge \\
& a \rightarrow (x' = x \wedge y' = y \wedge z' = z) \wedge \\
& \neg a \rightarrow (x' = y \wedge y' = z \wedge z' = y +_{[2^n]} 2)) \rightarrow \\
(y' = x' +_{[2^n]} 1 & \wedge z' = x' +_{[2^n]} 2)
\end{aligned}
$$

Here, the intended meaning is that variables x, y, and z are integers and $+_{[2^n]}$ indicates addition modulo 2^n.

When using first-order logic, part of our task is to specify the domain and the meaning of the symbols we are using.

First-Order Logic: Syntax

As with propositional logic, expressions in first-order logic are made up of sequences of symbols.

Symbols are divided into *logical symbols* and *non-logical symbols* or *parameters*.

Logical Symbols

- Parentheses: $($, $)$
- Propositional connectives: $\rightarrow$, $\neg$
- Variables: v_1, v_2, $\ldots$
- Universal quantifier: $\forall$

Parameters

- Equality symbol (optional): $=$
- Predicate symbols: e.g. $p(x)$, $x > y$
- Constant symbols: e.g. 0, *John*, π
- Function symbols: e.g. $f(x)$, $x + y$, $x +_{[2]} y$

Abbreviations

- Other propositional connectives: $\vee$, $\wedge$, $\leftrightarrow$
- Existential quantifier: $\exists\, x\; p(x) \Leftrightarrow \neg\forall\, x\; \neg p(x)$

Each predicate and function symbol has an associated *arity*, a natural number indicating how many arguments it takes.

Equality is a special predicate symbol of arity 2.

Constant symbols can also be thought of as functions of arity 0.

A *first-order language* must first specify its parameters.

First-Order Languages: Examples

Propositional Logic

- Equality: *no*
- Predicate symbols: A_1, A_2, ...
- Constant symbols: *none*
- Function symbols: *none*

Set Theory

- Equality: *yes*
- Predicate symbols: $\in$
- Constant symbols: $\emptyset$
- Function symbols: *none*

Elementary Number Theory

- Equality: *yes*
- Predicate symbols: $<$
- Constant symbols: 0
- Function symbols: S (successor), $+$, $\times$, exp

First-Order Logic: Terms

The first important concept on the way to defining well-formed formulas is that of *terms*.

For each function symbol f of arity n, we define a term-building operation $\mathcal{F}_f$:

$$\mathcal{F}_f(\alpha_1, \ldots, \alpha_n) = f\alpha_1, \ldots, \alpha_n$$

Note that we are using prefix notation to avoid ambiguity.

The set of *terms* is the set of expressions generated from the constant symbols and variables by the $\mathcal{F}_f$ operations.

Terms are expressions which name objects.

Theorem

The set of terms is freely generated from the set of variables and constant symbols by the $\mathcal{F}_f$ operations.

Atomic Formulas

An *atomic formula* is an expression of the form: $Pt_1, \ldots, t_n$ where P is a predicate symbol of arity n and t_1,...,t_n are terms.

If the language includes the equality symbol, we consider the equality symbol as a predicate of arity 2.

Formulas

We define the following formula-building operations:

- $\mathcal{E}_{\neg}(\alpha) = (\neg\alpha)$
- $\mathcal{E}_{\rightarrow}(\alpha, \beta) = (\alpha \rightarrow \beta)$
- $\mathcal{Q}_i(\alpha) = \forall\, v_i\ \alpha$

The set of *well-formed formulas* (*wffs* or just *formulas*) is the set of expressions generated from the atomic formulas by the operations $\mathcal{E}_{\neg}$, $\mathcal{E}_{\rightarrow}$, and $\mathcal{Q}_i$ $i = 1, 2, \ldots$.

This set is also freely generated.

Formula Examples

In the language of elementary number theory introduced above, which of the following are terms?

1. v_6 *yes*
2. $v_2 + v_3$ *no*
3. $+v_2v_3$ *yes*

atomic formulas?

1. $= exp + v_1 0 v_2 S v_3$ *yes*: $(v_1 + 0)^{v_2} = S(v_3)$
2. $\neg = v_2 v_3$ *no*

well-formed formulas?

1. $\neg = v_2 v_3$ *no*
2. $(\neg = v_2 v_3)$ *yes*: $v_2 \neq v_3$
3. $\times 0 v_1$ *no*
4. $\forall\, v_1\ = \times 0 v_1 v_1$ *yes*: $\forall\, v_1\ (0 \times v_1 = v_1)$

Free and Bound Variables

We define by recursion what it means for a variable x to *occur free* in a *wff* α:

- If α is an atomic formula, then x occurs free in α iff x occurs in α.
- x occurs free in $(\neg\alpha)$ iff x occurs free in α.

- x occurs free in $(\alpha \rightarrow \beta)$ iff x occurs free in α or in β.
- x occurs free in $\forall v_i \, \alpha$ iff x occurs free in α and $x \neq v_i$.

To make this definition precise, we would need to define a recursive function and make use of the recursion theorem and the fact that *wffs* are freely generated.

If $\forall v_i$ appears in α, then v_i is said to be *bound* in α.

> Note that a variable can both occur free and be bound in α. Because this can be confusing, we typically require the set of free and bound variables to be disjoint.

If no variable occurs free in a *wff* α, then α is a *sentence*.

First-Order Logic: Semantics

In propositional logic, the truth of a formula was determined by a *truth assignment* over the propositional symbols.

In first-order logic, we use a *model* (also known as a *structure*) to determine the truth of a formula.

A *signature* is a set of non-logical symbols (predicates, constants, and functions). Given a signature Σ, a model M of Σ consists of the following:

1. A nonempty set called the *domain* of M, written $dom(M)$. Elements of the domain are called elements of the model M.
2. A mapping from each constant c in Σ to an element c^M of M.
3. A mapping from each n-ary function symbol f in Σ to f^M, an n-ary function from $[dom(M)]^n$ to $dom(M)$.
4. A mapping from each n-ary predicate symbol p in Σ to $p^M \subseteq [dom(M)]^n$, an n-ary relation on the set $dom(M)$.

Example

Consider the signature corresponding to the language of set theory which has a single predicate symbol $\in$ and a single constant symbol $\emptyset$.

A possible model M for this signature has $dom(M) = \mathcal{N}$, the set of natural numbers, $\in^M = <$, and $\emptyset^M = 0$.

Now consider the sentence $\exists\, x\, \forall\, y\, \neg y \in x$.

What does this sentence mean in this model?

The translation of the sentence in the model M is that there is a natural number x such that no other natural number is smaller than x.

Is this sentence true in the model?

Since 0 has this property, the sentence is true in this model.

First-Order Logic: Semantics

We will often use a shorthand when discussing both signatures and models. The signature shorthand lists each symbol in the signature.

The model shorthand lists the domain and the interpretation of each symbol of the signature.

The signature for set theory can thus be described as ($\in$, $\emptyset$), and the above model as ($\mathcal{N}$, $<$, 0).

Given a model M, a *variable assignment* s is a function which assigns to each variable an element of M.

Given a wff ϕ, we say that M *satisfies* ϕ with s and write $\models_M \phi[s]$ if ϕ is true in the model M with variable assignment s.

To define this formally, we first define the extension $\overline{s} : T \to \mathit{dom}(M)$, a function from the set T of all terms into the domain of M:

1. For each variable x, $\overline{s}(x) = s(x)$.
2. For each constant symbol c, $\overline{s}(c) = c^M$.
3. If $t_1,\ldots,t_n$ are terms and f is an n-ary function symbol, then $\overline{s}(ft_1,\ldots,t_n) = f^M(\overline{s}(t_1),\ldots,\overline{s}(t_n))$.

The existence of a unique such extension $\overline{s}$ follows from the recursion theorem and the fact that the terms are freely generated.

Note that $\overline{s}$ depends on both s and M.

Atomic Formulas

1. $\models_M = t_1t_2[s]$ iff $\overline{s}(t_1) = \overline{s}(t_2)$.
2. For an n-ary predicate symbol P, $\models_M Pt_1,\ldots,t_n[s]$ iff $\langle\overline{s}(t_1),\ldots,\overline{s}(t_n)\rangle \in P^M$.

Other *wffs*

1. $\models_M (\neg\phi)[s]$ iff $\not\models_M \phi[s]$.
2. $\models_M (\phi \to \psi)[s]$ iff $\not\models_M \phi[s]$ or $\models_M \psi[s]$.
3. $\models_M \forall x\, \phi[s]$ iff $\models_M \phi[s(x|d)]$ for every $d \in \mathit{dom}(M)$.

$s(x|d)$ signifies the function which is the same as s everywhere except at x where its value is d.

Again, the well-formedness of this definition depends on the recursion theorem and the fact that *wffs* are freely generated.

Suppose Σ is a signature. A *Σ-formula* is a well-formed formula whose non-logical symbols are contained in Σ.

Let Γ be a set of Σ-formulas. We write $\models_M \Gamma[s]$ to signify that $\models_M \phi[s]$ for every $\phi \in \Gamma$.

If Γ is a set of Σ-formulas and ϕ is a Σ-formula, then Γ *logically implies* ϕ, written $\Gamma \models \phi$, iff for every model M of Σ and every variable assignment s, if $\models_M \Gamma[s]$ then $\models_M \phi[s]$.

We write $\psi \models \phi$ as an abbreviation for $\{\psi\} \models \phi$.

ψ and ϕ are *logically equivalent* (written $\psi \models\!\models \phi$) iff $\psi \models \phi$ and $\phi \models \psi$.

A Σ-formula ϕ is *valid*, written $\models \phi$ iff $\emptyset \models \phi$ (i.e. $\models_M \phi[s]$ for every M and s).

Examples

Suppose that P is a unary predicate and Q a binary predicate. Which of the following are true?

1. $\forall v_1\, Pv_1 \models Pv_2$ — *true*
2. $Pv_1 \models \forall v_1\, Pv_1$ — *false*
3. $\forall v_1\, Pv_1 \models \exists v_2\, Pv_2$ — *true*
4. $\exists x \forall y\, Qxy \models \forall y \exists x\, Qxy$ — *true*
5. $\forall x \exists y\, Qxy \models \exists y \forall x\, Qxy$ — *false*
6. $\models \exists x\, (Px \rightarrow \forall y\, Py)$ — *true*

Which models satisfy the following sentences?

1. $\forall x \forall y\, x = y$ — Models with exactly one element.
2. $\forall x \forall y\, Qxy$ — Models (A, R) where $R = A \times A$.
3. $\forall x \exists y\, Qxy$ — Models (A, R) where *dom*$(R) = A$.

Invariance of Truth Values

Theorem

Suppose s_1 and s_2 are variable assignments over a model M which agree at all variables (if any) which occur free in the *wff* ϕ. Then $\models_M \phi[s_1]$ iff $\models_M \phi[s_2]$.

Proof

The proof is by induction on well-formed formulas ϕ.

1. If ϕ is an atomic formula, then all variables in ϕ occur free. Thus s_1 and s_2 agree on all variables in ϕ. It follows that $\overline{s}_1(t) = \overline{s}_2(t)$ for each term t in ϕ (technically we should prove this by induction too). The result follows.
2. If ϕ is $(\neg\alpha)$ or $(\alpha \rightarrow \beta)$, the result is immediate from the inductive hypothesis.
3. Suppose $\phi = \forall\, x\, \psi$. The variables free in ϕ are the same as those free in ψ except for x. Thus, for any d in *dom*(M), $s_1(x|d)$ and $s_2(x|d)$ agree at all variables free in ψ. The result follows from the inductive hypothesis.

As a corollary of this theorem, we have that for sentences, satisfaction is independent of the variable assignment.

Definability Within a Model

Consider a fixed model M.

If ϕ is a formula whose free variables are among $v_1, \ldots, v_k$, and $a_1, \ldots, a_k$ are elements of M, then we write

$$\models_M \phi[[a_1, \ldots, a_k]]$$

to mean that M satisfies ϕ with some (and hence every) variable assignment s such that $s(v_i) = a_i$.

We can then associate with every such formula ϕ the k-ary relation:

$$\{\langle a_1, \ldots, a_k\rangle \mid \models_M \phi[[a_1, \ldots, a_k]]\}\,.$$

We say that this is the relation *defined by* ϕ in M.

In general, a k-ary relation on *dom*(M) is said to be *definable* in M iff there is a formula which defines it there.

Example

Consider the model $(\mathcal{N}, 0, S, +, \times)$.

- The ordering relation $\{\langle m, n\rangle \mid m < n\}$ is defined by $\exists\, v_3\ (v_1 + Sv_3 = v_2)$.
- For any natural number n, $\{n\}$ is definable. For example, $\{2\}$ is defined by $v_1 = SS0$.
- The set of primes is definable in $(\mathcal{N}, 0, S, +, \times)$:

  $$1 < v_1 \wedge \forall\, v_2\, \forall\, v_3\ (v_1 = v_2 \times v_3 \rightarrow v_2 = 1 \vee v_3 = 1).$$

  where $<$ can be defined as above, and 1 is an abbreviation for $S0$.

Notice that because there are uncountably many relations on $\mathcal{N}$ and only countably many possible defining formulas, some relations on $\mathcal{N}$ are not definable.