Last Week

Applying Propositional Logic

- Boolean Circuits
- Boolean Satisfiability (SAT)
- Binary Decision Diagrams (BDD’s)
Outline

- Computability and Decidability
- First-Order Logic: Motivation
- First-Order Logic: Syntax
- First-Order Logic: Semantics
- Definability

Sources:

Sections 1.7 through 2.3 of Enderton.

N. J. Cutland. *Computability*.

Computability

The important notion of *computability* relies on a formal model of computation.

Many formal models have been proposed:

1. General recursive functions defined by means of an equation calculus (Gödel-Herbrand-Kleene)
2. $\lambda$-definable functions (Church)
3. $\mu$-recursive functions and partial recursive functions (Gödel-Kleene)
4. Functions computable by finite machines known as Turing machines (Turing)
5. Functions defined from canonical deduction systems (Post)
6. Functions given by certain algorithms over a finite alphabet (Markov)
7. Universal Register Machine-computable functions (Shepherdson-Sturgis)

**Fundamental Result**

All of these (and many other) models of computation are equivalent. That is, they give rise to the same class of functions.
Computability and Decidability

All of these models are equivalent to what can be achieved by a computer with any standard programming language, given arbitrary (but finite) time and memory.

Church’s Thesis

A notion known as Church’s thesis states that all models of computation are either equivalent to or less powerful than those just described.

We will accept Church’s thesis and thus define a function to be *computable* if we can describe precisely (using any model of computation) how to compute it. Such a description will be called an *effective procedure*. 
Computability and Decidability

All of these models are equivalent to what can be achieved by a computer with any standard programming language, given arbitrary (but finite) time and memory.

Church’s Thesis

A notion known as Church’s thesis states that all models of computation are either equivalent to or less powerful than those just described.

We will accept Church’s thesis and thus define a function to be *computable* if we can describe precisely (using any model of computation) how to compute it. Such a description will be called an *effective procedure*.

Decidability

Given a universal set $U$, a set $S \subseteq U$ is decidable if there exists a computable function $f : U \rightarrow \{F, T\}$ such that $f(x) = T$ iff $x \in S$. 
Computability and Decidability

All of these models are equivalent to what can be achieved by a computer with any standard programming language, given arbitrary (but finite) time and memory.

Church’s Thesis

A notion known as Church’s thesis states that all models of computation are either equivalent to or less powerful than those just described.

We will accept Church’s thesis and thus define a function to be computable if we can describe precisely (using any model of computation) how to compute it. Such a description will be called an effective procedure.

Decidability

Given a universal set $U$, a set $S \subseteq U$ is decidable if there exists a computable function $f : U \rightarrow \{F, T\}$ such that $f(x) = T$ iff $x \in S$.

Decidability of $W$

Earlier, we presented an algorithm which, given any expression $\alpha$ determines whether the expression is well-formed. Thus, the set $W$ of well-formed formulas is decidable.
Decidability

Some decidable sets

- For a given finite set of wffs $\Sigma$, the set of all tautological consequences of $\Sigma$ (i.e. $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.
Decidability

Some decidable sets

- For a given finite set of wffs $\Sigma$, the set of all tautological consequences of $\Sigma$ (i.e. $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.

  The truth table algorithm given earlier decides $\Sigma \models \alpha$. 
Decidability

Some decidable sets

- For a given finite set of wffs $\Sigma$, the set of all tautological consequences of $\Sigma$ (i.e. $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.
  
  The truth table algorithm given earlier decides $\Sigma \models \alpha$.

- The set of tautologies is decidable.
Decidability

Some decidable sets

- For a given finite set of wffs $\Sigma$, the set of all tautological consequences of $\Sigma$ (i.e., $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.

  The truth table algorithm given earlier decides $\Sigma \models \alpha$.

- The set of tautologies is decidable.

  The set of tautologies is just the set of tautological consequences of the empty set.
Decidability

Some decidable sets

- For a given finite set of *wffs* $\Sigma$, the set of all *tautological consequences* of $\Sigma$ (i.e. $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.

  The truth table algorithm given earlier decides $\Sigma \models \alpha$.

- The set of tautologies is decidable.

  The set of tautologies is just the set of tautological consequences of the empty set.

Existence of undecidable sets

A simple argument shows the existence of undecidable sets of expressions: an algorithm is completely determined by its finite description. Thus, there are only countably many effective procedures. But there are uncountably many sets of expressions.

Why?
Decidability

Some decidable sets

- For a given finite set of \( \text{wffs} \Sigma \), the set of all \textit{tautological consequences} of \( \Sigma \) (i.e. \( \{ \alpha \mid \Sigma \models \alpha \} \)) is decidable.
  
  The truth table algorithm given earlier decides \( \Sigma \models \alpha \).

- The set of tautologies is decidable.
  
  The set of tautologies is just the set of tautological consequences of the empty set.

Existence of undecidable sets

A simple argument shows the existence of undecidable sets of expressions: an algorithm is completely determined by its finite description. Thus, there are only countably many effective procedures. But there are uncountably many sets of expressions.

Why?

The set of expressions is countably infinite. Therefore, its power set is uncountable.
Semi-Decidability

Suppose we wish to determine whether $\Sigma \models \alpha$ where $\Sigma$ is infinite. In general, this is not decidable.

But we can obtain a weaker result:

A set $A$ is semi-decidable (or effectively enumerable) if there is an effective procedure which lists, in some order, every member of $A$.

Note that if $A$ is infinite, then the procedure will never finish, but every member of $A$ must appear in the list after some finite amount of time.
Semi-Decidability

Suppose we wish to determine whether $\Sigma \models \alpha$ where $\Sigma$ is infinite. In general, this is not decidable.

But we can obtain a weaker result:

A set $A$ is semi-decidable (or effectively enumerable) if there is an effective procedure which lists, in some order, every member of $A$.

Note that if $A$ is infinite, then the procedure will never finish, but every member of $A$ must appear in the list after some finite amount of time.

**Theorem**

A set $A$ of expressions is effectively enumerable iff there is an effective procedure which, given any expression $\alpha$, produces the answer “yes” iff $\alpha \in A$. 
Semi-Decidability

Suppose we wish to determine whether $\Sigma \models \alpha$ where $\Sigma$ is infinite. In general, this is not decidable.

But we can obtain a weaker result:

A set $A$ is semi-decidable (or effectively enumerable) if there is an effective procedure which lists, in some order, every member of $A$.

Note that if $A$ is infinite, then the procedure will never finish, but every member of $A$ must appear in the list after some finite amount of time.

**Theorem**

A set $A$ of expressions is effectively enumerable iff there is an effective procedure which, given any expression $\alpha$, produces the answer “yes” iff $\alpha \in A$.

**Proof**

If $A$ is effectively enumerable, then we simply enumerate its members and check each one to see if it is equivalent to $\alpha$. If it is, we return “yes” and stop. Otherwise, we keep going. Thus, if $\alpha \in A$, the procedure produces “yes”. If $\alpha \notin A$, the procedure runs forever.
Proof, continued

On the other hand, suppose that we have an effective procedure $P$ which produces “yes” iff $\alpha \in A$. To produce an enumeration of $A$, we proceed as follows. First enumerate all expressions:

$$\epsilon_1, \epsilon_2, \epsilon_3, \ldots$$

Then proceed as follows.

- Break the procedure $P$ into a finite number of “steps”.
- Run $P$ on $\epsilon_1$ for 1 step.
- Run $P$ on $\epsilon_1$ for 2 steps, and then run $P$ on $\epsilon_2$ for 2 steps.
- ... 
- Run $P$ on each of $\epsilon_1, \ldots, \epsilon_n$ for $n$ steps each
- ... 

If at any time, the procedure $P$ produces “yes”, then we list the expression which produced “yes” and continue.

This procedure will eventually enumerate all members of $A$. \qed
Semi-Decidability

Theorem

A set is decidable iff both it and its complement (with respect to a given universal set) are effectively enumerable.
Semi-Decidability

Theorem

A set is decidable iff both it and its complement (with respect to a given universal set) are effectively enumerable.

Proof

Alternate between running the procedure for the set and the procedure for its complement. One of them will eventually produce “yes”.
Semi-Decidability

Theorem

A set is decidable iff both it and its complement (with respect to a given universal set) are effectively enumerable.

Proof

Alternate between running the procedure for the set and the procedure for its complement. One of them will eventually produce “yes”.

Properties of decidable and semi-decidable sets

Decidable sets are closed under union, intersection, and complement.

Semi-decidable sets are closed under union and intersection.
Semi-Decidability

Theorem

If \( \Sigma \) is an effectively enumerable set of \( \text{wffs} \), then the set of tautological consequences of \( \Sigma \) is effectively enumerable.
Semi-Decidability

Theorem

If $\Sigma$ is an effectively enumerable set of wffs, then the set of tautological consequences of $\Sigma$ is effectively enumerable.

Proof

Consider an enumeration of the elements of $\Sigma$:

$$\sigma_1, \sigma_2, \sigma_3, \ldots$$

By the compactness theorem, $\Sigma \models \alpha$ iff $\{\sigma_1, \ldots, \sigma_n\} \models \alpha$ for some $n$.

Hence, it is sufficient to successively test:

- $\emptyset \models \alpha$
- $\{\sigma_1\} \models \alpha$
- $\{\sigma_1, \sigma_2\} \models \alpha$
- $\ldots$

If any of these conditions is met (each of which is decidable), the answer is “yes”.
Semi-Decidability

Theorem

If $\Sigma$ is an effectively enumerable set of wffs, then the set of tautological consequences of $\Sigma$ is effectively enumerable.

Proof (continued)

This demonstrates that there is an effective procedure that, given any wff $\alpha$, will output “yes” iff $\alpha$ is a tautological consequence of $\Sigma$.

Thus, the set of tautological consequences of $\Sigma$ is effectively enumerable. □
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.
Example
Example

2-bit counter property specification:
\[ z_1 \leftrightarrow \neg x_1 \land z_0 \leftrightarrow x_0 \land y_1 \leftrightarrow (x_1 \oplus x_0) \land y_0 \leftrightarrow \neg x_0 \]

\( n \)-bit counter specification requires a formula of size \( O(n) \).

Using first-order logic, we can express the specification using a formula whose size is constant for all \( n \):

\[ z = x +_{[2^n]} 2 \land y = x +_{[2^n]} 1 \]

Here, the intended meaning is that variables \( x \), \( y \), and \( z \) range over the set \([0..2^n - 1]\) and \( +_{[2^n]} \) indicates addition modulo \( 2^n \).

When using first-order logic, part of our task is to specify 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: →, ¬
- Variables: $v_1, v_2, \ldots$
- Universal quantifier: ∀

**Parameters**
- Equality symbol (optional): =
- Predicate symbols: e.g. $p(x), x > y$
- Constant symbols: e.g. 0, John, π
- Function symbols: e.g. $f(x), x + y, x + [2] y$
First-Order Logic: Syntax

Abbreviations

- Other propositional connectives: $\lor$, $\land$, $\leftrightarrow$
- Existential quantifier: $\exists x \ p(x) \iff \neg \forall x \ \neg p(x)$

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: $A_1, A_2, \ldots$
- Constant symbols: none
- Function symbols: none

Set Theory

- Equality: yes
- Predicate symbols: $\in$
- Constant symbols: $\emptyset$
- Function symbols: none
First-Order Languages: Examples

Elementary Number Theory

- Equality: \textit{yes}
- Predicate symbols: \textless
- 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 $F_f$:

$$F_f(\alpha_1, \ldots, \alpha_n) = f\alpha_1, \ldots, \alpha_n$$

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 $F_f$ 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 $F_f$ operations.
First-Order Logic: Formulas

Atomic Formulas

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

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

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 $Q_i$ $i = 1, 2, \ldots$

This set is also freely generated.
Formula Examples

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

1. $u_6$
In the language of elementary number theory introduced above, which of the following are terms?

1. $v_6$  yes
2. $v_2 + v_3$
Formula Examples

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

1. \( v_6 \) \hspace{1cm} yes
2. \( v_2 + v_3 \) \hspace{1cm} no
3. \( +v_2 v_3 \)
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. $exp + v_10v_2Sv_3$
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. $=exp + v_1 0v_2 Sv_3$  yes: $(v_1 + 0)^v_2 = S(v_3)$
2. $= \neg v_2v_3$
Formula Examples

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

1. $v_6$ \hspace{1cm} yes
2. $v_2 + v_3$ \hspace{1cm} no
3. $+v_2 v_3$ \hspace{1cm} yes

atomic formulas?

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

well-formed formulas?

1. $= v_2 v_3$
Formula Examples

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

1. \( v_6 \) \hspace{1cm} yes
2. \( v_2 + v_3 \) \hspace{1cm} no
3. \( +v_2 v_3 \) \hspace{1cm} yes

atomic formulas?

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

well-formed formulas?

1. \( \neg = v_2 v_3 \) \hspace{1cm} no
2. \( (\neg = v_2 v_3) \)
Formula Examples

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

1. \( v_6 \) \hspace{1cm} yes
2. \( v_2 + v_3 \) \hspace{1cm} no
3. \( +v_2 v_3 \) \hspace{1cm} yes

atomic formulas?

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

well-formed formulas?

1. \( \neg = v_2 v_3 \) \hspace{1cm} no
2. \( (\neg = v_2 v_3) \) \hspace{1cm} yes: \( v_2 \neq v_3 \)
3. \( \times 0 v_1 \)
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. \( = exp + v_10v_2Sv_3 \) yes: \((v_1 + 0)^v_2 = S(v_3)\)
2. \( = v_2v_3 \) no

well-formed formulas?

1. \( = v_2v_3 \) no
2. \( (\neg = v_2v_3) \) yes: \( v_2 \neq v_3 \)
3. \( \times0v_1 \) no
4. \( \forall v_1 = \times0v_1v_1 \)
Formula Examples

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

1. \( v_6 \) \hspace{1cm} yes
2. \( v_2 + v_3 \) \hspace{1cm} no
3. \( +v_2v_3 \) \hspace{1cm} yes

atomic formulas?

1. \[ = \exp + v_10v_2Sv_3 \] \hspace{1cm} yes: \( (v_1 + 0)^{v_2} = S(v_3) \)
2. \( \neg = v_2v_3 \) \hspace{1cm} no

well-formed formulas?

1. \( \neg = v_2v_3 \) \hspace{1cm} no
2. \( \neg = v_2v_3 \) \hspace{1cm} yes: \( v_2 \neq v_3 \)
3. \( \times 0v_1 \) \hspace{1cm} no
4. \( \forall v_1 = \times 0v_1v_1 \) \hspace{1cm} 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 $\alpha$:

- If $\alpha$ is an atomic formula, then $x$ occurs free in $\alpha$ iff $x$ occurs in $\alpha$.
- $x$ occurs free in $(-\alpha)$ iff $x$ occurs free in $\alpha$.
- $x$ occurs free in $(\alpha \to \beta)$ iff $x$ occurs free in $\alpha$ or in $\beta$.
- $x$ occurs free in $\forall v_i \alpha$ iff $x$ occurs free in $\alpha$ 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 $\alpha$, then $v_i$ is said to be bound in $\alpha$.

Note that a variable can both occur free and be bound in $\alpha$. 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 $\alpha$, then $\alpha$ 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 $\Sigma$, a model $M$ of $\Sigma$ consists of the following:

1. A nonempty set called the domain of $M$, written $\text{dom}(M)$. Elements of the domain are called elements of the model $M$.

2. A mapping from each constant $c$ in $\Sigma$ to an element $c^M$ of $M$.

3. A mapping from each $n$-ary function symbol $f$ in $\Sigma$ to $f^M$, an $n$-ary function from $[\text{dom}(M)]^n$ to $\text{dom}(M)$.

4. A mapping from each $n$-ary predicate symbol $p$ in $\Sigma$ to $p^M \subseteq [\text{dom}(M)]^n$, an $n$-ary relation on the set $\text{dom}(M)$. 
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 $\text{dom}(M) = \mathcal{N}$, the set of natural numbers, $\in^M = <$, and $\emptyset^M = 0$.

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

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 $x$.

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

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 $(\mathcal{N}, <, 0)$.
First-Order Logic: Semantics

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

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

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

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

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

Note that $\bar{s}$ depends on both $s$ and $M$. 
First-Order Logic: Semantics

Atomic Formulas

1. $\models_M t_1 t_2[s]$ iff $\overline{s}(t_1) = \overline{s}(t_2)$.

2. For an $n$-ary predicate symbol $P$,
   $\models_M P t_1, \ldots, t_n[s]$ 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. $\models_M (\phi \rightarrow \psi)[s]$ 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 $d \in \text{dom}(M)$.

$s(x|d)$ 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.
Logical Definitions

Suppose $\Sigma$ is a signature. A $\Sigma$-formula is a well-formed formula whose non-logical symbols are contained in $\Sigma$.

Let $\Gamma$ be a set of $\Sigma$-formulas. We write $\models M \Gamma[s]$ to signify that $\models M \phi[s]$ for every $\phi \in \Gamma$.

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

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

$\psi$ and $\phi$ are logically equivalent (written $\psi \equiv \phi$) iff $\psi \models \phi$ and $\phi \models \psi$.

A $\Sigma$-formula $\phi$ 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. $\forall v_1 P v_1 \models P v_2$
Examples

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

1. $\forall v_1 \ P v_1 \models P v_2 \quad \text{true}$
2. $P v_1 \models \forall v_1 \ P v_1$
Examples

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

1. $\forall v_1 \ P v_1 \models P v_2$  
   true

2. $P v_1 \models \forall v_1 \ P v_1$  
   false

3. $\forall v_1 \ P v_1 \models \exists v_2 \ P v_2$
Examples

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

1. $\forall v_1 \ P v_1 \models P v_2$ \hspace{1cm} true
2. $P v_1 \models \forall v_1 \ P v_1$ \hspace{1cm} false
3. $\forall v_1 \ P v_1 \models \exists v_2 \ P v_2$ \hspace{1cm} true
4. $\exists x \ \forall y \ Q x y \models \forall y \ \exists x \ Q x y$
Examples

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

1. $\forall v_1 P v_1 \models P v_2$ \text{ true}
2. $P v_1 \models \forall v_1 P v_1$ \text{ false}
3. $\forall v_1 P v_1 \models \exists v_2 P v_2$ \text{ true}
4. $\exists x \forall y Q x y \models \forall y \exists x Q x y$ \text{ true}
5. $\forall x \exists y Q x y \models \exists y \forall x Q x y$
Examples

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

1. $\forall v_1 P v_1 \models P v_2$ \hspace{1cm} true
2. $P v_1 \models \forall v_1 P v_1$ \hspace{1cm} false
3. $\forall v_1 P v_1 \models \exists v_2 P v_2$ \hspace{1cm} true
4. $\exists x \forall y Q xy \models \forall y \exists x Q xy$ \hspace{1cm} true
5. $\forall x \exists y Q xy \models \exists y \forall x Q xy$ \hspace{1cm} false
6. $\models \exists x (P x \to \forall y P y)$
Examples

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

1. $\forall v_1 P v_1 \models P v_2$  \hspace{1cm} \text{true}
2. $P v_1 \models \forall v_1 P v_1$ \hspace{1cm} \text{false}
3. $\forall v_1 P v_1 \models \exists v_2 P v_2$ \hspace{1cm} \text{true}
4. $\exists x \forall y Q x y \models \forall y \exists x Q x y$ \hspace{1cm} \text{true}
5. $\forall x \exists y Q x y \models \exists y \forall x Q x y$ \hspace{1cm} \text{false}
6. $\models \exists x (P x \to \forall y P y)$ \hspace{1cm} \text{true}

Which models satisfy the following sentences?

1. $\forall x \forall y x = y$
Examples

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

1. $\forall v_1 \ P v_1 \models P v_2$  
   true
2. $P v_1 \models \forall v_1 \ P v_1$  
   false
3. $\forall v_1 \ P v_1 \models \exists v_2 \ P v_2$  
   true
4. $\exists x \ \forall y \ Q xy \models \forall y \ \exists x \ Q xy$  
   true
5. $\forall x \ \exists y \ Q xy \models \exists y \ \forall x \ Q xy$  
   false
6. $\models \exists x \ (P x \rightarrow \forall y \ P y)$  
   true

Which models satisfy the following sentences?

1. $\forall x \ \forall y \ x = y$  
   Models with exactly one element.
2. $\forall x \ \forall y \ Q xy$
Examples

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

1. $\forall v_1 \ P v_1 \models P v_2$  \text{true}
2. $P v_1 \models \forall v_1 \ P v_1$  \text{false}
3. $\forall v_1 \ P v_1 \models \exists v_2 \ P v_2$  \text{true}
4. $\exists x \ \forall y \ Q x y \models \forall y \ \exists x \ Q x y$  \text{true}
5. $\forall x \ \exists y \ Q x y \models \exists y \ \forall x \ Q x y$  \text{false}
6. $\models \exists x \ (P x \rightarrow \forall y \ P y)$  \text{true}

Which models satisfy the following sentences?

1. $\forall x \ \forall y \ x = y$  Models with exactly one element.
2. $\forall x \ \forall y \ Q x y$  Models $(A, R)$ where $R = A \times A$.
3. $\forall x \ \exists y \ Q x y$
Examples

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

1. $\forall v_1 \, P v_1 \models P v_2$ \hspace{1cm} true
2. $P v_1 \models \forall v_1 \, P v_1$ \hspace{1cm} false
3. $\forall v_1 \, P v_1 \models \exists v_2 \, P v_2$ \hspace{1cm} true
4. $\exists x \, \forall y \, Q x y \models \forall y \, \exists x \, Q x y$ \hspace{1cm} true
5. $\forall x \, \exists y \, Q x y \models \exists y \, \forall x \, Q x y$ \hspace{1cm} false
6. $\models \exists x \, (P x \rightarrow \forall y \, P y)$ \hspace{1cm} true

Which models satisfy the following sentences?

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

28-i
Invariance of Truth Values

Theorem

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

Proof

The proof is by induction on well-formed formulas $\phi$.

1. If $\phi$ is an atomic formula, then all variables in $\phi$ occur free. Thus $s_1$ and $s_2$ agree on all variables in $\phi$. It follows that $s_1(t) = s_2(t)$ for each term $t$ in $\phi$ (technically we should prove this by induction too). The result follows.

2. If $\phi$ 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 $\phi$ are the same as those free in $\psi$ except for $x$. Thus, for any $d$ in $\text{dom}(\mathcal{M})$, $s_1(x|d)$ and $s_2(x|d)$ agree at all variables free in $\psi$. 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 $\phi$ 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 $\phi$ with some (and hence every) variable assignment $s$ such that $s(v_i) = a_i$.

We can then associate with every such formula $\phi$ 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 $\phi$ in $M$.

In general, a $k$-ary relation on $\operatorname{dom}(M)$ 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 \((\mathcal{N}, 0, S, +, \times)\):
  \[ 1 < v_1 \land \forall v_2 \forall v_3 \ (v_1 = v_2 \times v_3 \rightarrow v_2 = 1 \lor v_3 = 1). \]
  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.