something or other
Syntax
Components of Syntax

Words

\( a, b, g, p \)

Terms

\( g(a,a) \)

Sentences

\( \forall x. (p(x) \Rightarrow p(g(x,x))) \)
Words are strings of letters, digits, and occurrences of the underscore character.

*Variables* begin with characters from the end of the alphabet (from *u* through *z*).

\[ u, v, w, x, y, z \]

*Constants* begin with digits or letters from the beginning of the alphabet (from *a* through *t*).

\[ a, b, c, 123, \text{comp225}, \text{barack\_obama} \]
Object constants represent objects.

jo, stanford, usa, 2345

Function constants represent functions.

father, mother, age, plus, times

Relation constants represent relations.

knows, loves
The *arity* of a function constant or a relation constant is the number of arguments it takes.

*Unary* function or relation constant - 1 argument

*Binary* function or relation constant - 2 arguments

*Ternary* function or relation constant - 3 arguments

*n*-ary function or relation constant - *n* arguments
A signature consist of a set of object constants, a set of function constants, and a set of relation constants together with a specification of arity for the function constants and relation constants.

Object Constants: \(a, b\)

Unary Function Constant: \(f\)
Binary Function Constant: \(g\)

Unary Relation Constant: \(p\)
Binary Relation Constant: \(q\)
A *term* is either a variable, an object constant, or a functional term (defined shortly).

Terms represent objects.

Terms are analogous to noun phrases in natural language.
A *functional term* is an expression formed from an $n$-ary function constant and $n$ terms enclosed in parentheses and separated by commas.

\[
\begin{align*}
  f(a) \\
  f(x) \\
  g(a, y)
\end{align*}
\]

Functional terms are terms and so can be nested.

\[
g(f(a), g(y,a))
\]
Three types of sentences in Herbrand Logic:

Relational sentences - analogous to the simple sentences in natural language

Logical sentences - analogous to the logical sentences in natural language

Quantified sentences - sentences that express the significance of variables
A *relational sentence* is an expression formed from an $n$-ary relation constant and $n$ terms enclosed in parentheses and separated by commas.

$$q(a, f(a))$$

Relational sentences are *not* terms and *cannot* be nested in terms or relational sentences.

No! $q(a, q(a, y))$  No!
Logical sentences in Herbrand Logic are analogous to those in Propositional Logic.

\(-q(a,b)\)
\((p(a) \land p(b))\)
\((p(a) \lor p(b))\)
\((q(x,y) \Rightarrow q(y,x))\)
\((q(x,y) \Leftrightarrow q(y,x))\)
Quantified Sentences

Universal sentences assert facts about all objects.

\[(\forall x. (p(x) \Rightarrow q(x, f(x))))\]

Existential sentence assert the existence of objects with given properties.

\[(\exists x. (p(x) \land q(x, f(x))))\]

Quantified sentences can be nested within other sentences.

\[(\forall x. p(x)) \lor (\exists x. q(x, f(x)))\]

\[\forall x. (\exists y. q(f(x), y))\]
Parentheses can be removed when precedence allows us to reconstruct sentences correctly.

Precedence relations same as in Propositional Logic with quantifiers being of *higher* precedence than logical operators.

\[
\forall x. p(x) \Rightarrow q(x, x) \rightarrow (\forall x. p(x)) \Rightarrow q(x, x)
\]

\[
\exists x. p(x) \land q(x, x) \rightarrow (\exists x. p(x)) \land q(x, x)
\]
Semantics
The *Herbrand base* for a Herbrand language is the set of all ground relational sentences that can be formed from the vocabulary of the language.
Object Constants: \( a, b \)
Unary Relation Constant: \( p \)
Binary Relation Constant Constant: \( q \)

Herbrand Base:

\[ \{ p(a), p(b), q(a,a), q(a,b), q(b,a), q(b,b) \} \]
Object Constants: $a$
Unary Function Constant: $f$
Unary Relation Constant: $p$

Herbrand Base:

\[ \{ p(a), p(f(a)), p(f(f(a))), \ldots \} \]
A truth assignment is an association between ground atomic sentences and the truth values true or false. As with Propositional Logic, we use 1 as a synonym for true and 0 as a synonym for false.

\[
\begin{align*}
p(a)^i &= 1 \\
p(b)^i &= 0 \\
q(a,a)^i &= 1 \\
q(a,b)^i &= 0 \\
q(b,a)^i &= 1 \\
q(b,b)^i &= 0
\end{align*}
\]
All other notions are defined the same as in Relational Logic.

The main difference is that now we have truth assignments that are infinitely large and there are infinitely many of them. This means that it is no longer possible in general to determine properties and relationships like logical entailment in finite time.
Example - Peano Arithmetic
In *Peano Arithmetic*, we are concerned with all of the natural numbers, not just a finite subset, and functions do not wrap around as in Modular Arithmetic.

\[
\begin{align*}
0+0&=0 \\
1+0&=1 \\
2+0&=2 \\
3+0&=3 \\
0+1&=1 \\
1+1&=2 \\
2+1&=3 \\
3+1&=5 \\
0+2&=2 \\
1+2&=3 \\
2+2&=4 \\
3+2&=6 \\
0+3&=3 \\
1+3&=4 \\
2+3&=5 \\
3+3&=7 \\
\cdots &\quad \cdots &\quad \cdots &\quad \cdots &\quad \cdots \\
\end{align*}
\]
Object Constants: 0, 1, 2, …

Ground Terms: 0, 1, 2, …
Possible Representations

Object Constants: 0, 1, 2, …

Ground Terms: 0, 1, 2, …

Object Constant: 0
Unary Function Constant: s

Ground Terms: 0, s(0), s(s(0)), …
Object Constant: 0

Unary Function Constant: $s$

Binary Relation Constant:
   \textit{same} - the first and second arguments are identical

Ternary Relation Constant:
   \textit{plus} - the third argument is the sum of the first two
   \textit{times} - third argument is the product of the first two
Enumerating ground relational data impossible

$$same(0,0) \quad plus(0,0,0) \quad times(0,0,0)$$
$$\neg same(0,s(0)) \quad \neg plus(0,0,s(0)) \quad \neg times(0,0,s(0))$$
$$\neg same(0,s(s(0))) \quad \neg plus(0,0,s(s(0))) \quad \neg times(0,0,s(s(0)))$$

… … …

Solution - write logical and quantified sentences
Definition:

\[ \forall x. \text{same}(x, x) \]

\[ \forall x. (\neg \text{same}(0, s(x)) \land \neg \text{same}(s(x), 0)) \]

\[ \forall x. \forall y. (\neg \text{same}(x, y) \Rightarrow \neg \text{same}(s(x), s(y))) \]
Definition:

$$\forall x. \text{same}(x, x)$$

$$\forall x. (\neg \text{same}(0, s(x)) \land \neg \text{same}(s(x), 0))$$

$$\forall x. \forall y. (\neg \text{same}(x, y) \implies \neg \text{same}(s(x), s(y)))$$

Examples:

$$\text{same}(0, 0)$$

$$\text{same}(s(0), s(0))$$

$$\text{same}(s(s(0)), s(s(0)))$$

$$\ldots$$
Definition:

\[ \forall x. \text{same}(x,x) \]

\[ \forall x. (\neg \text{same}(0,s(x)) \land \neg \text{same}(s(x),0)) \]

\[ \forall x. \forall y. (\neg \text{same}(x,y) \implies \neg \text{same}(s(x), s(y))) \]

Examples:

\[ \neg \text{same}(0,s(0)) \]

\[ \text{same}(0,0) \]

\[ \text{same}(s(0),s(0)) \]

\[ \neg \text{same}(0,s(s(0))) \]

\[ \text{same}(s(s(0)),s(s(0))) \]

\[ \neg \text{same}(0,s(s(s(0)))) \]

\[ \text{same}(s(s(s(0))),s(s(s(0)))) \]

\[ \neg \text{same}(0,s(s(s(s(0))))) \]

\[ \text{same}(s(s(s(s(0)))),s(s(s(s(0))))) \]
Definition:

\[ \forall x. \text{same}(x,x) \]

\[ \forall x. (\neg \text{same}(0, s(x)) \land \neg \text{same}(s(x), 0)) \]

\[ \forall x. \forall y. (\neg \text{same}(x, y) \Rightarrow \neg \text{same}(s(x), s(y))) \]

Examples:

\( \text{same}(0,0) \)

\( \text{same}(s(0), s(0)) \)

\( \text{same}(s(s(0)), s(s(0))) \)

\( \ldots \)

\( \neg \text{same}(0, s(0)) \)

\( \neg \text{same}(0, s(s(0))) \)

\( \ldots \)

\( \neg \text{same}(s(0), s(s(0))) \)

\( \neg \text{same}(s(0), s(s(s(0)))) \)

\( \ldots \)
Addition

Identity:
\[ \forall y. \text{plus}(0, y, y) \]

Successor:
\[ \forall x. \forall y. \forall z. (\text{plus}(x, y, z) \Rightarrow \text{plus}(s(x), y, s(z))) \]

Functionality:
\[ \forall x. \forall y. \forall z. \forall w. (\text{plus}(x, y, z) \land \text{plus}(x, y, w) \Rightarrow \text{same}(z, w)) \]
Identity:

\[ \forall y. \text{times}(0, y, 0) \]

Successor:

\[ \forall x. \forall y. \forall z. (\text{times}(x, y, z) \land \text{plus}(y, z, w) \Rightarrow \text{times}(s(x), y, w)) \]

Functionality:

\[ \forall x. \forall y. \forall z. \forall w. (\text{times}(x, y, z) \land \text{times}(x, y, w) \Rightarrow \text{same}(z, w)) \]
A polynomial equation is an algebraic expression composed using only addition and multiplication and exponentiation with fixed exponents.

\[ 3x^2 + 2y = 2z^3 \]

A natural Diophantine equation is a polynomial in which the values of variables are restricted to natural numbers.

\[ x = 2 \]
\[ y = 2 \]
\[ z = 2 \]
Diophantine equation:

\[3x^2 = 1\]

Diophantine equation in Peano Arithmetic:

\[\forall x. \forall y. \forall z. \quad (times(x,x,y) \land times(s(s(s(0))),y,z) \Rightarrow same(z,s(0)))\]

Diophantine equation with “syntactic sugar”:

\[\forall x. \forall y. \forall z.(x*x=y \land 3*y=z \Rightarrow z=1)\]

\[\forall x. \forall y. \forall z.(3*x*x=1)\]
Example - Linked Lists
Flat Lists:

$$[a, b, c, d]$$

Nested Lists:

$$[a, [a, b], b, [c, d], d]$$

Linked List:
Example

Representation as Term

\[ \text{cons}(a, \text{cons}(b, \text{cons}(c, \text{cons}(d, \text{nil})))) \]
Object Constants: $a, b, c, d, nil$

Binary Function Constant: $\text{cons}$

Binary Relation Constants: $\text{member, among}$

Ternary Relation Constant: $\text{append}$
Example:

\( \text{member}(b, \text{cons}(a,\text{cons}(b,\text{cons}(c,\text{nil})))) \)

Definition:

\[
\forall x. \forall y. \text{member}(x, \text{cons}(x, y))
\]

\[
\forall x. \forall y. \forall z. (\text{member}(x, z) \implies \text{member}(x, \text{cons}(y, z)))
\]
Example:
\[
\text{append}(\text{cons}(a,\text{cons}(b,\text{nil})), \\
\text{cons}(c,\text{cons}(d,\text{nil})), \\
\text{cons}(a,\text{cons}(b,\text{cons}(c,\text{cons}(d,\text{nil})))))
\]

Definition:
\[
\forall x. \forall y. \text{append}(\text{nil}, y, y) \\
\forall x. \forall y. \forall z. \forall w. (\text{append}(x, y, w) \\
\quad \Rightarrow \text{append}(\text{cons}(x, y), z, \text{cons}(x, w)))
\]
Example:

\[ \text{among}(c,\text{cons}(a, \text{cons}(\text{cons}(b,\text{cons}(c,nil)), \text{cons}(d,nil)))) \]

Definition:

\[
\forall x. \text{among}(x,x) \\
\forall x. \forall y. \forall z. (\text{among}(x,y) \lor \text{among}(x,z) \Rightarrow \text{among}(x,\text{cons}(y,z)))
\]
Example - Pseudo English
Good Sentences:
Mary likes Pat.
Mary likes Pat and Quincy.
Pat and Quincy like Mary.

Bad Sentences:
Mary Pat likes.
Likes and Mary Pat Quincy.
Backus Naur Form

<sentence> ::= <np> <vp>

<np> ::= <noun>
<np> ::= <noun> "and" <noun>
<vp> ::= <verb> <np>

<noun> ::= "mary" | "pat" | "puincy"
<verb> ::= "like" | "likes"
Logical Grammar

\[ np(x) \land vp(y) \land \text{append}(x,y,z) \implies \text{sentence}(z) \]

\[ \text{noun}(x) \implies np(x) \]
\[ \text{noun}(x) \land \text{noun}(y) \land \text{append}(x,\text{and},z) \land \text{append}(z,y,w) \implies \text{np}(w) \]

\[ \text{verb}(x) \land np(y) \land \text{append}(x,y,z) \implies \text{vp}(z) \]

\[ \text{noun}(\text{mary}) \]
\[ \text{noun}(\text{pat}) \]
\[ \text{noun}(\text{quincy}) \]

\[ \text{verb}(\text{like}) \]
\[ \text{verb}(\text{likes}) \]
Logical Grammar

\[ np(x) \land vp(y) \land append(x,y,z) \Rightarrow sentence(z) \]

\[ noun(x) \Rightarrow np(x) \]
\[ noun(x) \land noun(y) \land append(x,\text{and},z) \land append(z,y,w) \Rightarrow np(w) \]

\[ verb(x) \land np(y) \land append(x,y,z) \Rightarrow vp(z) \]

noun(mary)
noun(pat)
noun(quincy)

verb(like)
verb(likes)
Logical Grammar

\[ np(x) \land vp(y) \land append(x,y,z) \implies sentence(z) \]

\[ noun(x) \implies np(x) \]
\[ noun(x) \land noun(y) \land append(x,\text{and},z) \land append(z,y,w) \implies np(w) \]

\[ verb(x) \land np(y) \land append(x,y,z) \implies vp(z) \]

noun(mary)
noun(pat)
noun(quincy)

verb(like)
verb(likes)
Logical Grammar

\[ np(x) \land vp(y) \land append(x,y,z) \implies sentence(z) \]

\[ noun(x) \implies np(x) \]

\[ noun(x) \land noun(y) \land append(x,\text{and},z) \land append(z,y,w) \implies np(w) \]

\[ verb(x) \land np(y) \land append(x,y,z) \implies vp(z) \]

\[ noun(mary) \]
\[ noun(pat) \]
\[ noun(quincy) \]

\[ verb(like) \]
\[ verb(likes) \]
Logical Grammar

\[ np(x) \land vp(y) \land append(x, y, z) \Rightarrow sentence(z) \]

\[ noun(x) \Rightarrow np(x) \]
\[ noun(x) \land noun(y) \land append(x, and, z) \land append(z, y, w) \Rightarrow np(w) \]

\[ verb(x) \land np(y) \land append(x, y, z) \Rightarrow vp(z) \]

\[ noun(mary) \]
\[ noun(pat) \]
\[ noun(quinc) \]

\[ verb(like) \]
\[ verb(likes) \]
Examples

Sentences:
√ Mary likes Pat.
√ Mary likes Pat and Quincy.
√ Pat and Quincy like Mary.

Not Sentences:
× Mary Pat likes.
× Likes and Mary Pat Quincy.
Sentences:
  Mary likes Pat.
  Mary likes Pat and Quincy.
  Pat and Quincy like Mary.

Allowed but not sentences in natural English:
  Mary likes Pat.
  Pat and Quincy likes Mary.

How can we enforce subject-verb number agreement?
Augmented Logic Grammar

\[
np(x,w) \land vp(y,w) \land \text{append}(x,y,z) \Rightarrow \text{sentence}(z)
\]

\[
noun(x) \Rightarrow np(x,0)
\]

\[
noun(x) \land noun(y) \land \text{append}(x,\text{and},z) \land \text{append}(z,y,w) \Rightarrow np(w,1)
\]

\[
verb(x,w) \land np(y,v) \land \text{append}(x,y,z) \Rightarrow vp(z,w)
\]

noun(mary)
noun(pat)
noun(quincy)

verb(like,1)
verb(likes,0)
Augmented Logic Grammar

\[ np(x,w) \land vp(y,w) \land append(x,y,z) \Rightarrow sentence(z) \]

\[ noun(x) \Rightarrow np(x,0) \]
\[ noun(x) \land noun(y) \land append(x,\text{and},z) \land append(z,y,w) \Rightarrow np(w,1) \]

\[ verb(x,w) \land np(y) \land append(x,y,z) \Rightarrow vp(z,w) \]

\[ noun(mary) \]
\[ noun(pat) \]
\[ noun(quincy) \]

\[ verb(like,1) \]
\[ verb(likes,0) \]
Augmented Logic Grammar

\[ np(x,w) \land vp(y,w) \land append(x,y,z) \implies sentence(z) \]

\[ noun(x) \implies np(x,0) \]
\[ noun(x) \land noun(y) \land append(x,\text{and},z) \land append(z,y,w) \implies np(w,1) \]

\[ verb(x,w) \land np(y) \land append(x,y,z) \implies vp(z,w) \]

\[ noun(mary) \]
\[ noun(pat) \]
\[ noun(quincy) \]

\[ verb(like,1) \]
\[ verb(likes,0) \]
Augmented Logic Grammar

\[ np(x,w) \land vp(y,w) \land append(x,y,z) \implies sentence(z) \]

\[ noun(x) \implies np(x,0) \]
\[ noun(x) \land noun(y) \land append(x,\text{and},z) \land append(z,y,w) \implies np(w,1) \]

\[ verb(x,w) \land np(y) \land append(x,y,z) \implies vp(z,w) \]

\[ noun(\text{mary}) \]
\[ noun(\text{pat}) \]
\[ noun(\text{quincy}) \]

\[ verb(\text{like},1) \]
\[ verb(\text{likes},0) \]
Augmented Logic Grammar

\[ np(x,w) \land vp(y,w) \land append(x,y,z) \Rightarrow sentence(z) \]

\[ noun(x) \Rightarrow np(x,0) \]
\[ noun(x) \land noun(y) \land append(x,and,z) \land append(z,y,w) \Rightarrow np(w,1) \]

\[ verb(x,w) \land np(y) \land append(x,y,z) \Rightarrow vp(z,w) \]

- noun(mary)
- noun(pat)
- noun(quincy)

- verb(like,1)
- verb(likes,0)
Example - Metalevel Logic
Basic idea: represent expressions in Propositional Logic as terms in Herbrand Logic, write Herbrand sentences to define basic concepts of Propositional Logic, prove metatheorems.

NB: We can extend to Herbrand Logic as well. The formalization is messier, and some nasty problems need to be handled (notably paradoxes).
Object Constants (propositions)

\( p, q, r \)
Object Constants (propositions)

\[ p, q, r \]

Function constants

\[ \neg p \quad if(p,q) \]
\[ \land(p,q) \quad iff(p,q) \]
\[ \lor(p,q) \]
## Syntactic Metavocabulary

### Object Constants (propositions)
- $p$, $q$, $r$

### Function constants
- $\text{not}(x)$
- $\text{if}(x,y)$
- $\text{and}(x,y)$
- $\text{iff}(x,y)$
- $\text{or}(x,y)$

### Relation Constants
- $\text{proposition}(x)$
- $\text{implication}(x)$
- $\text{negation}(x)$
- $\text{biconditional}(x)$
- $\text{conjunction}(x)$
- $\text{sentence}(x)$
- $\text{disjunction}(x)$
Syntactic Metadefinitions

proposition(p)

proposition(q)

proposition(r)

negation(not(x)) ⇔ sentence(x)

conjunction(and(x,y)) ⇔ sentence(x) ∧ sentence(y)

disjunction(or(x,y)) ⇔ sentence(x) ∧ sentence(y)

implication(if(x,y)) ⇔ sentence(x) ∧ sentence(y)

equivalence(iff(x,y)) ⇔ sentence(x) ∧ sentence(y)

sentence(x) ⇔

  proposition(x) ∨ negation(x) ∨ conjunction(x) ∨
  disjunction(x) ∨ implication(x) ∨ biconditional(x)
Rules of Inference

And Introduction:

$$\forall x. (\text{sentence}(x) \land \text{sentence}(y) \Rightarrow ai(x,y,\text{and}(x,y)))$$

And Elimination:

$$\forall x. (\text{sentence}(x) \land \text{sentence}(y) \Rightarrow ae(\text{and}(x,y),x))$$
$$\forall x. (\text{sentence}(x) \land \text{sentence}(y) \Rightarrow ae(\text{and}(x,y),y))$$
Validity of Axiom Schemes:

\[ \text{valid}(\text{or}(x,\neg x)) \iff \text{sentence}(x) \]

Soundness:

\[ \text{proves}(x,y) \iff \text{entails}(x,y) \]

Deduction Theorem:

\[ \text{proves}(\text{and}(x,y),z) \iff \text{proves}(x,\text{implies}(y,z)) \]
In similar fashion, we can axiomatize Herbrand Logic in Herbrand Logic. But with some care to avoid paradoxes (sentences that are both true and false) and self-contradictory terms.

Paradox:

This sentence is false.

Self-contradictory terms:

This set of all sets that do not contain themselves