Introduction to Logic

Term Logic

Michael Genesereth
Computer Science Department
Stanford University

Motivation

Finite Worlds

- *n* rows x *n* columns in Friends, Goldrush, Minefinder
- Finite Graphs
- University Students
- Population of a state or country

Countable Worlds

- Integers - 1, 2, 3, 4, ...
- Strings - "adbyug78377bh", ...
- Sequences - [], [a], [b], [a,a], [a,b], [b,a], [b,b], [a,a,a], ...
- Sets - {}, {a}, {b}, {a,b}, {{a},{b}}, {{a},{a,b}}, ...

Possibilities

Infinite Relational Logic - Infinite Vocabulary

$a1, a2, a3, \ldots$

Term Logic - Structured Terms

$0, s(0), s(s(0)), s(s(s(0))), \ldots$

$a, b, pair(a,a), pair(b,a), pair(b,b), pair(a, pair(a,b)), \ldots$

$a, b, set(), set(a), set(b), set(a,b), set(set(a),set(a,b)), \ldots$

Programme

Today
- Syntax
- Semantics
- Properties and Relationships
- Examples

Next Time
- Fitch Proofs with Induction

Next Week
- Equality
- Review

Syntax

Words

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).

$$a, b, c, \mathit{123}, \mathit{father}, \mathit{mother}, \mathit{comp225}, \mathit{barack\_obama}$$

Constants

Object constants (*symbols*) represent objects.

joe, *stanford*, *france*, 2345

Function constants (*constructors*) represent functions.

successor, pair, set

Relation constants (*predicates*) represent relations.

knows, *loves*

Arity

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

Signatures

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

Terms

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 (e.g. *France*, *the set of 2 and 3*)

Functional Terms

A *functional term* is an expression consisting of an n-ary function constant and n terms enclosed in parentheses and separated by commas.

$$f(a)$$
$$f(x)$$
$$g(a, y)$$

Functional terms are terms and so *can* be nested*.

$$g(f(a), g(y,a))$$

* *unlike relational sentences*

Sentences

Three types of sentences in Term 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

Relational Sentences

A *relational sentence* is an expression formed from an n-ary relation constant and n terms enclosed in parentheses and separated by commas.

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

Reminder: Relational sentences are *not* terms and *cannot* be nested inside terms or relational sentences.

No! $q(a,q(a,y))$ No!

Logical Sentences

Logical sentences in Term Logic are analogous to those in Propositional Logic (except with functional terms).

$$(\neg q(a,f(a)))$$

$$(p(a) \wedge p(f(a)))$$

$$(p(a) \vee p(f(a)))$$

$$(q(x,f(a)) \Rightarrow q(f(a),x))$$

$$(q(x,f(a)) \Leftrightarrow q(f(a),x))$$

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.

$$(\exists x.(p(x) \wedge q(x,f(x))))$$

Quantified sentences can be nested within other sentences.

$$(\forall x.p(x)) \vee (\exists x.q(x,f(x)))$$
$$(\forall x.(\exists y.q(f(x),y)))$$

Parentheses

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.

$$\forall x.p(x) \Rightarrow q(x,x) \rightarrow (\forall x.p(x)) \Rightarrow q(x,x)$$
$$\exists x.p(x) \wedge q(x,x) \rightarrow (\exists x.p(x)) \wedge q(x,x)$$

Semantics

Herbrand Universe and Herbrand Base

The *Herbrand universe* for a Functional language is the set of all *ground terms* that can be formed from the vocabulary of the language.

The *Herbrand base* for a Functional language is the set of all *ground relational sentences* that can be formed from the vocabulary of the language.

Example Without Functions

Object Constants: a, b
Unary Relation Constant: p
Binary Relation Constant: q

Herbrand Universe:

$$\{a, b\}$$

Herbrand Base:

$$\{p(a), p(b), q(a,a), q(a,b), q(b,a), q(b,b)\}$$

Example With Functions

Object Constants: a
Unary Function Constant: f
Unary Relation Constant: p

Herbrand Universe:

$$\{a, f(a), f(f(a)), \ldots\}$$ *Infinite!!!*

Herbrand Base:

$$\{p(a), p(f(a)), p(f(f(a))), \ldots\}$$ *Infinite!!!*

Truth Assignments

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*.

$p(a)^i = 1$	$q(a,a)^i = 1$
$p(b)^i = 0$	$q(a,b)^i = 0$
$p(f(a))^i = 1$	$q(a,f(a))^i = 0$
$p(f(b))^i = 0$	$q(a,f(b))^i = 1$
$p(f(f(a)))^i = 0$	$q(b,f(a))^i = 0$
$p(f(f(b)))^i = 0$	$q(b,f(b))^i = 1$
...	...

Infinite!!!

Everything Else

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.

Bad News: It is no longer possible in general to determine logical entailment and other properties with truth tables.

Good News: In many cases, logical entailment can be established with finite proofs.

Example - Whole Numbers

Whole Numbers

Entities (natural numbers together with 0):

$$0, 1, 2, 3, 4, \ldots$$

Successor:

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow \ldots$$

Less Than (transitive closure of successor):

$$\begin{array}{ccc} 0 < 1 & 1 < 2 & \ldots \\ 0 < 2 & 1 < 3 & \ldots \\ 0 < 3 & 1 < 4 & \ldots \\ \ldots & \ldots & \ldots \end{array}$$

Possible Representations

Object Constants: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...

Ground Terms: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...

Possible Representations

Object Constants: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...
Ground Terms: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ...

Object Constant: 0
Unary Function Constant: s
Ground Terms: 0, $s(0)$, $s(s(0))$, …

NB: spelling matters in our standard notation for numbers

- We do not write as a, b, c, d, ...
- We write as 0, 1, 2,..., 9, [1,0], [1,1], [1,2], ..., [1,0,0], ...
- Arithmetic operations take advantage of this

Signature

Object Constant: 0

Unary Function Constant: *s*

Binary Relation Constants:

same - the first and second arguments are identical
succ - the first argument immediately precedes second
less - the first argument less than or equal to second

Axiomatization

Enumerating ground relational data impossible

$$
\begin{array}{lll}
same(0,0) & \neg succ(0,0) & \neg less(0,0) \\
\neg same(0,s(0)) & succ(0,s(0)) & less(0,s(0)) \\
\neg same(0,s(s(0))) & \neg succ(0,s(s(0))) & less(0,s(s(0))) \\
\ldots & \ldots & \ldots
\end{array}
$$

Solution - write logical and quantified sentences

Same

Definition:

$$\forall x. same(x,x)$$

$$\forall x. (\neg same(0, s(x)) \wedge \neg same(s(x), 0))$$

$$\forall x. \forall y. (\neg same(x,y) \Rightarrow \neg same(s(x), s(y)))$$

Same

Definition:

$$\forall x.same(x,x)$$

$$\forall x.(\neg same(0,s(x)) \wedge \neg same(s(x),0))$$

$$\forall x.\forall y.(\neg same(x,y) \Rightarrow \neg same(s(x), s(y)))$$

Examples:

$same(0,0)$

$same(s(0),s(0))$

$same(s(s(0)),s(s(0)))$

$\ldots$

Same

Definition:

$$\forall x.same(x,x)$$

$$\forall x.(\neg same(0,s(x)) \wedge \neg same(s(x),0))$$

$$\forall x.\forall y.(\neg same(x,y) \Rightarrow \neg same(s(x), s(y)))$$

Examples:

$same(0,0)$	$\neg same(0,s(0))$	$\neg same(s(0),0)$
$same(s(0),s(0))$	$\neg same(0,s(s(0)))$	$\neg same(s(s(0)),0)$
$same(s(s(0)),s(s(0)))$	...	...
...		

Same

Definition:

$$\forall x.same(x,x)$$

$$\forall x.(\neg same(0,s(x)) \wedge \neg same(s(x),0))$$

$$\forall x.\forall y.(\neg same(x,y) \Rightarrow \neg same(s(x), s(y)))$$

Examples:

$same(0,0)$	$\neg same(0,s(0))$	$\neg same(s(0),0)$
$same(s(0),s(0))$	$\neg same(0,s(s(0)))$	$\neg same(s(s(0)),0)$
$same(s(s(0)),s(s(0)))$	$\dots$	$\dots$
$\dots$	$\neg same(s(0),s(s(0)))$	$\neg same(s(s(0)),s(0))$
	$\neg same(s(0),s(s(s(0))))$	$\neg same(s(s(s(0))),s(0))$
	$\dots$	$\dots$

Successor

Positives:

$$\forall y.succ(x,s(x))$$

Functionality:

$$\forall x.\forall y.\forall z.(succ(x,y) \wedge succ(x,z) \Rightarrow same(y,z))$$

or

$$\forall x.\forall y.\forall z.(succ(x,y) \wedge \neg same(y,z) \Rightarrow \neg succ(x,z))$$

Less Than

Successor:

$$\forall x.\forall y.(succ(x,y) \Rightarrow less(x,y))$$

Transitivity:

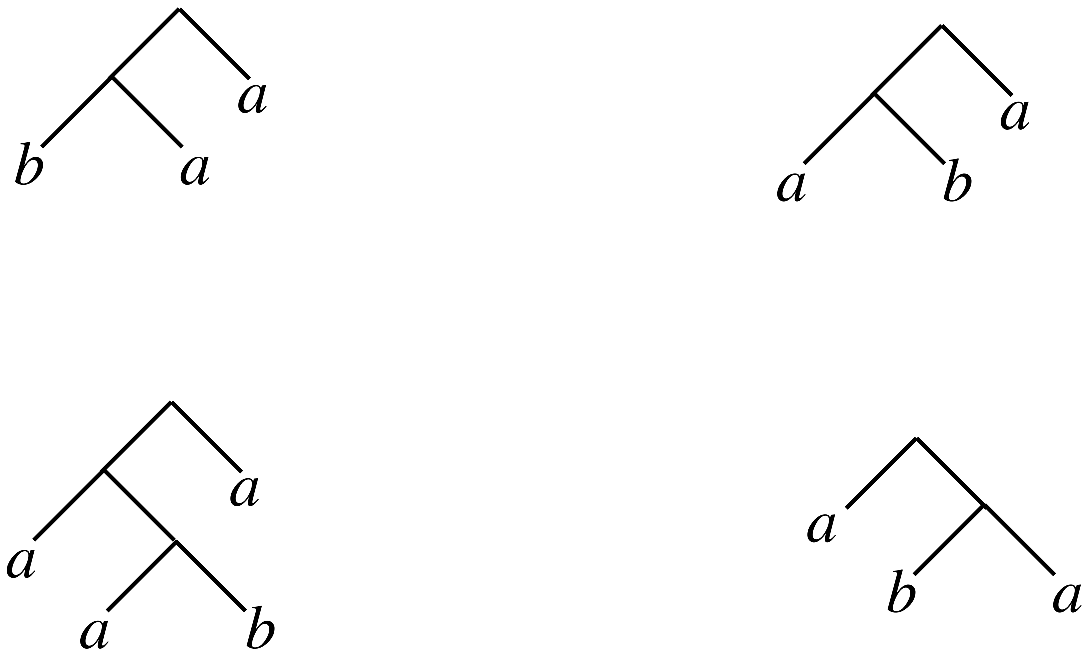
$$\forall x.\forall y.\forall z.(less(x,y) \wedge less(y,z) \Rightarrow less(x,z))$$

Irreflexivity:

$$\forall x.\neg less(x,x)$$

Example - Trees

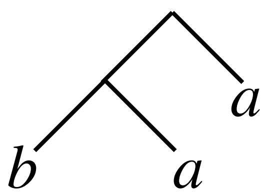
Trees



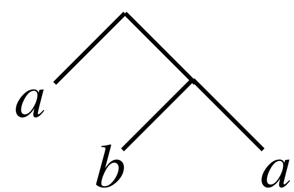
Tree Vocabulary

Object constants: *a*, *b*

Binary function constants: *cons*



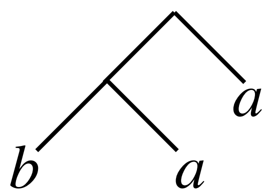
$cons(cons(b,a),a)$



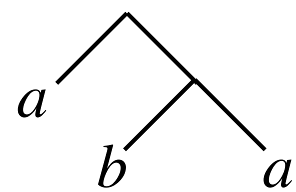
$cons(a,cons(b,a))$

Tree Vocabulary

Object constants: a, b
Unary function constants: $cons$



$$cons(cons(b, a), a)$$



$$cons(a, cons(b, a))$$

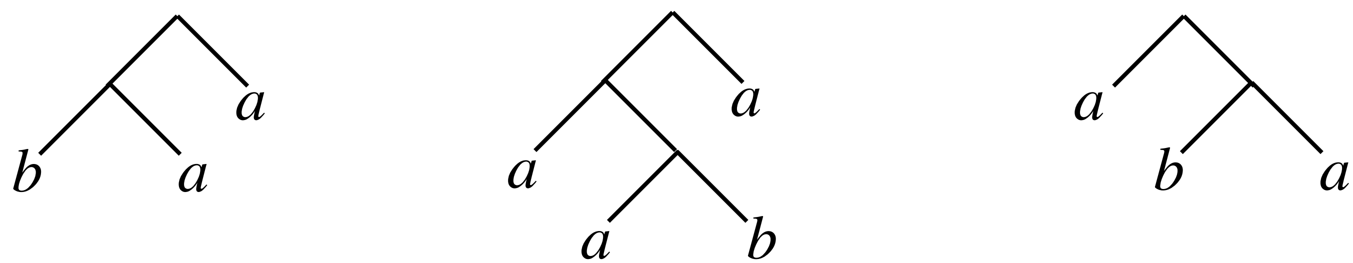
Unary relation constants: $symmetric$, $uniform$, ...
Binary relation constant: $subtree$, $congruent$, $mirror$, ...

Congruence

Two trees are *congruent* if and only if they have the *same shape*. (Labels on leaf nodes irrelevant.)

Examples:

Non-Examples:



Definition

Congruence of atomic trees

$$congruent(a, a)$$
$$congruent(a, b)$$
$$congruent(b, a)$$
$$congruent(b, b)$$

Congruence of compound trees:

$$\forall u.\forall v.\forall x.\forall y.(congruent(cons(u, v), cons(x, y)) \Leftrightarrow congruent(u, x) \ \wedge congruent(v, y))$$

Non-Congruence of mixed trees:

$$\forall x.\forall y.(\neg congruent(a, cons(x, y)) \wedge \neg congruent(cons(x, y), a))$$
$$\forall x.\forall y.(\neg congruent(b, cons(x, y)) \wedge \neg congruent(cons(x, y), b))$$

Example - Linked Lists

Linked Lists

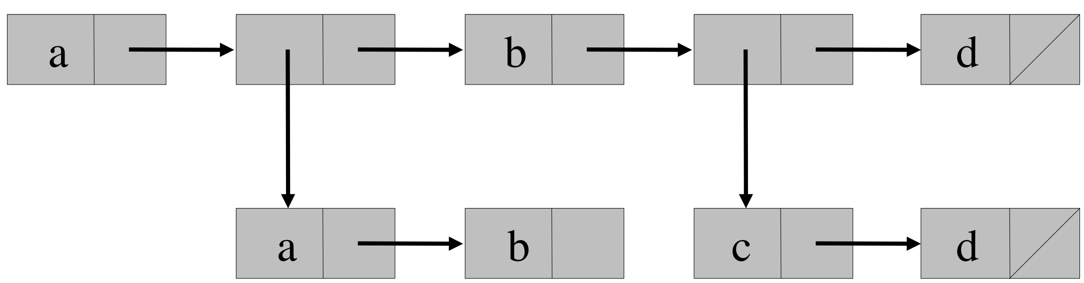
Flat Lists:

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

Nested Lists:

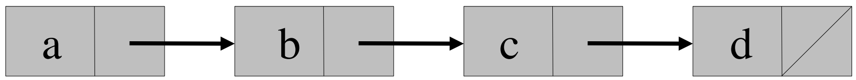
$$[a, [a, b], b, [c, d], d]$$

Linked List:



Representation

Example:



Representation as a functional term:

$$cons(a, cons(b, cons(c, cons(d, nil))))$$

Signature

Object Constants: a, b, c, d, nil

Binary Function Constant: *cons*

Binary Relation Constant: *member*
Ternary Relation Constant: *append*

$$member(b, [a, b, c])$$
$$append([a, b], [c, d], [a, b, c, d])$$

Membership

Example: $member(b, [a, b, c])$

$$member(b, cons(a, cons(b, cons(c, nil))))$$

Definition:

$$\forall x.\forall y.member(x, cons(x, y)))$$
$$\forall x.\forall y.\forall z.(member(x, z) \Rightarrow member(x, cons(y, z)))$$

What else do we need?

Concatenation

Example: $append([a,b],[c,d],[a,b,c,d])$

$$\begin{aligned} &append(cons(a,cons(b,nil)),\\ &\qquad cons(c,cons(d,nil)),\\ &\qquad cons(a,cons(b,cons(c,cons(d,nil))))) \end{aligned}$$

Definition :

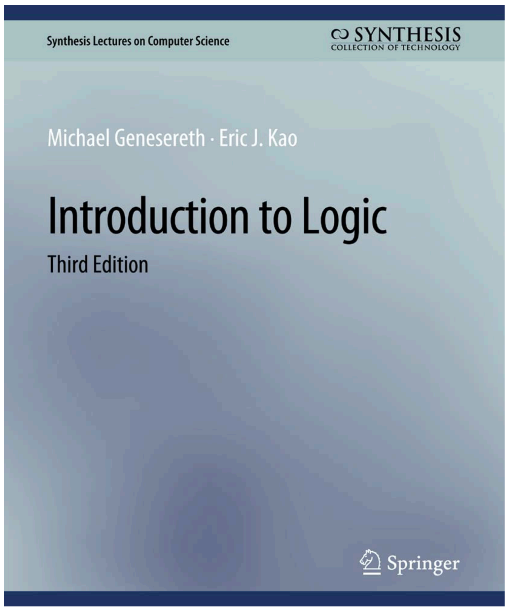
$$\forall y.append(nil,y,y)$$

$$\begin{aligned} &\forall x.\forall y.\forall z.\forall w.(append(y,z,w)\\ &\qquad \Rightarrow append(cons(x,y),z,cons(x,w))) \end{aligned}$$

What else do we need?

Example - Metalevel Logic

Metalevel Logic

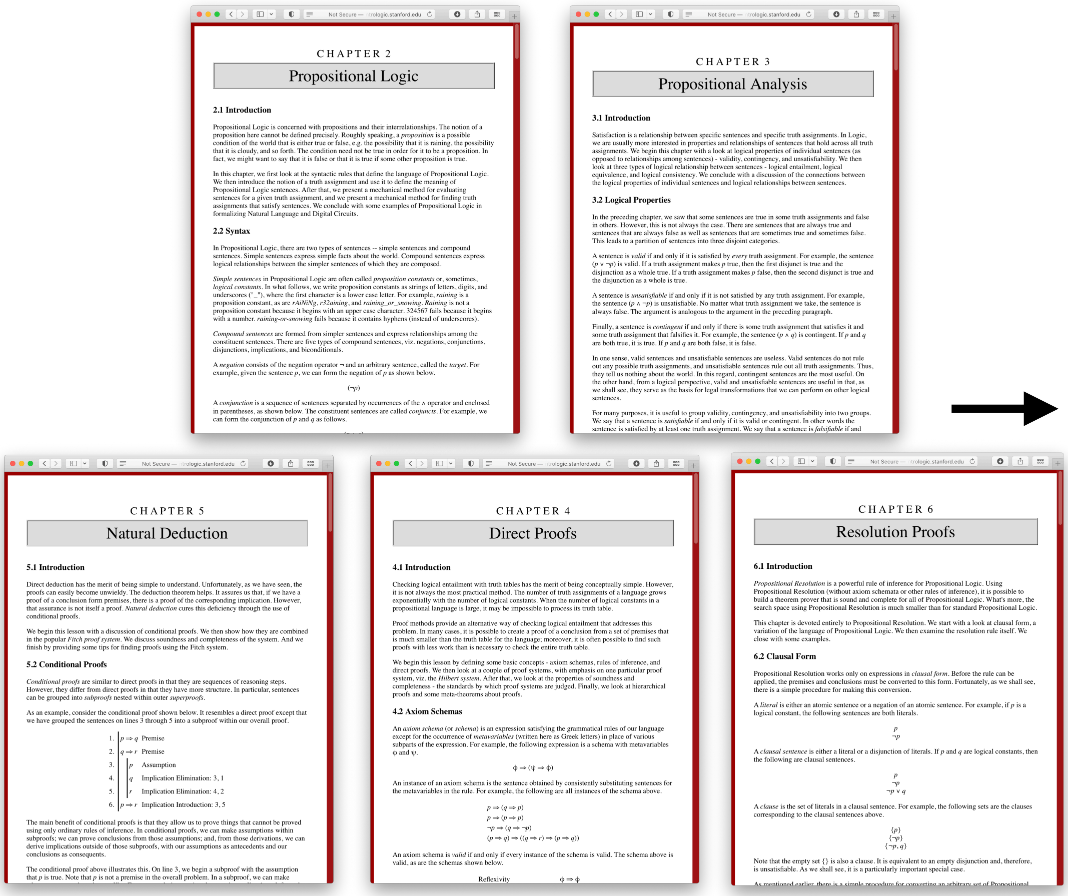


$proposition(p)$
$proposition(q)$
$proposition(r)$

$negation(not(x)) \Leftrightarrow sentence(x)$
$conjunction(and(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$disjunction(or(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$implication(if(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$biconditional(iff(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$

$sentence(x) \Leftrightarrow$
$proposition(x) \vee negation(x) \vee conjunction(x) \vee$
$disjunction(x) \vee implication(x) \vee biconditional(x)$

Propositional Logic in Term Logic



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)*
biconditional(iff(x,y)) ⇔ *sentence(x)* ∧ *sentence(y)*

sentence(x) ⇔
proposition(x) ∨ *negation(x)* ∨ *conjunction(x)* ∨
disjunction(x) ∨ *implication(x)* ∨ *biconditional(x)*

Basic Idea

(1) Represent Propositional Logic *sentences* as *terms* in Term Logic.

$$p \wedge \neg q \text{ represented as } and(p,not(q))$$

(2) Write Term Logic sentences to define the syntax and semantics of Propositional Logic.

$$conjunction(and(p,not(q)))$$

(3) Create Term Logic proofs of Propositional Logic metatheorems (e.g. soundness, completeness, deduction theorem, and so forth).

$$\forall x.\forall y.(entails(x,y) \Rightarrow proves(x,y))$$

Syntactic Metavocabulary

Object Constants (representing *propositions*):

p, q, r

Syntactic Metavocabulary

Object Constants (representing propositions):

p, *q*, *r*

Function constants (representing logical operators):

not(x)	*if(x,y)*	
and(x,y)	*iff(x,y)*	*These are **terms**!!*
or(x,y)		

Syntactic Metavocabulary

Object Constants (representing propositions):

p, *q*, *r*

Function constants (representing logical operators):

not(x)
and(*x*,*y*)
or(*x*,*y*)
if(x,y)
iff(x,y)

*These are **terms**!!*

Unary Relation Constants (properties of sentences):

proposition(*x*)
negation(*x*)
conjunction(*x*)
disjunction(*x*)
implication(*x*)
biconditional(*x*)
sentence(*x*)

Syntactic Metadefinitions

$proposition(p)$
$proposition(q)$
$proposition(r)$

$negation(not(x)) \Leftrightarrow sentence(x)$
$conjunction(and(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$disjunction(or(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$implication(if(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$biconditional(iff(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$

$sentence(x) \Leftrightarrow$
$\quad proposition(x) \vee negation(x) \vee conjunction(x) \vee$
$\quad disjunction(x) \vee implication(x) \vee biconditional(x)$

Semantic Metavocabulary

Unary Relation Constants (properties of sentences):
valid(x) - validity
contingent(x) - contingency
unsatisfiable(x) - unsatisfiability

Binary Relation Constants (relations among sentences):
equivalent(x,y) - logical equivalence
entails(x,y) - logical entailment
consistent(x,y) - consistency

We also need to talk about truth assignments in order to define these notions. Doable but messy; skipping here.

Semantic Metatheorems

Validity of Axiom Schemata:

$$valid(or(x,not(x))) \Leftrightarrow sentence(x)$$

Equivalence and Entailment:

$$equivalent(x,y) \Leftrightarrow entails(x,y) \wedge entails(y,x)$$

Deduction Theorem:

$$entails(and(x,y),z) \Leftrightarrow entails(x,if(y,z))$$

Rules of Inference

And Introduction:

$$\forall x.\forall y.(sentence(x) \wedge sentence(y) \Leftrightarrow ai(x,y,and(x,y)))$$

And Elimination:

$$\forall x.\forall y.(sentence(x) \wedge sentence(y) \Leftrightarrow ae(and(x,y),x))$$
$$\forall x.\forall y.(sentence(x) \wedge sentence(y) \Leftrightarrow ae(and(x,y),y))$$

More Metatheorems

Soundness:

$$\forall x.\forall y.(proves(x,y) \Rightarrow entails(x,y))$$

Completeness:

$$\forall x.\forall y.(entails(x,y) \Rightarrow proves(x,y))$$

Term Logic in Term Logic

Can we define Term Logic in Term Logic?

Basic idea: represent Term Logic expressions as terms in Term Logic, write sentences to define syntax and semantics, prove metatheorems.

Syntactic Metavocabulary

NB: We need terms to represent *functional terms* and *relational sentences*.

$$p(a,f(a)) \qquad relsent(p,a,funterm(f,a)))$$

NB: We need *constants* in our language to refer to *variables* in the language we are describing.

$$\forall y.p(y,f(y)) \qquad forall(ny,relsent(p,ny,funterm(f,ny)))$$

Syntactic Metadefinitions

$obconst(a)$
$funconst(f)$
$relconst(r)$
$variable(nx)$

$functionalterm(funterm(w,x)) \Leftrightarrow funconst(w) \wedge term(x)$
$relationalsentence(relsent(w,x)) \Leftrightarrow relconst(w) \wedge term(x)$

$negation(not(x)) \Leftrightarrow sentence(x)$
$conjunction(and(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$disjunction(or(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$implication(if(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$biconditional(iff(x,y)) \Leftrightarrow sentence(x) \wedge sentence(y)$
$universal(forall(v,x) \Leftrightarrow variable(v) \wedge sentence(x)$
$universal(exists(v,x) \Leftrightarrow variable(v) \wedge sentence(x)$

Cardinality Problem

In formalizing Propositional Logic, we *can* talk about truth assignments. The Herbrand base is always finite, and so there are only finitely many truth assignments.

In formalizing Term Logic, things are more difficult. The Herbrand base can be infinite (though it is always *countable*). However, the number of truth assignments can be *uncountable*. Unfortunately, we have only countably many terms!

Term Logic in Another Logic

*Can we define the semantics of Term Logic in some **other** logic?*

Good News / Bad News: First-Order Logic (FOL) allows for uncountable universes and so in principle can be used. Unfortunately, FOL theories with infinite universes have *nonstandard models* (unintended models that *cannot be excluded*).

NB: FOL is *weaker* than Term Logic. Some notions that can be defined exactly in Term Logic cannot be defined in FOL without allowing nonstandard models, e.g. Peano Arithmetic, transitive closure.

Functional Logic in Another Logic

Can we define the semantics of Term Logic in some other logic?

Good News / Bad News: First-Order Logic (FOL) allows for uncountable universes and so in principle can be used. Unfortunately, FOL theories with infinite universes have *nonstandard models* (unintended models that *cannot be excluded*).

Good News / Bad News: Second-Order Logic (SOL) allows us to eliminate these nonstandard models, but it is more complicated and there is no complete proof procedure.

Self-Referential Logic

Can we use this "metalevel" approach to relate the truth of sentences described in a metalanguage to sentences describing those sentences?

Truth Predicate

Can we use this "metalevel" approach to relate the truth of sentences described in a metalanguage to sentences describing those sentences?

Example: If so, can we define a *truth predicate* that allows us to say whether or not a sentence is true?

$$\forall x.\forall y.(true(relsent(p,x,y)) \Leftrightarrow p(x,y))$$

Beliefs

Can we use this "metalevel" approach to relate the truth of sentences described in a metalanguage to sentences describing those sentences?

Example: Can we use our truth predicate to formalize the truth of people's beliefs, beliefs about those beliefs, etc.?

$$\forall x.(believes(john, x) \Leftrightarrow true(x))$$

Disinformation

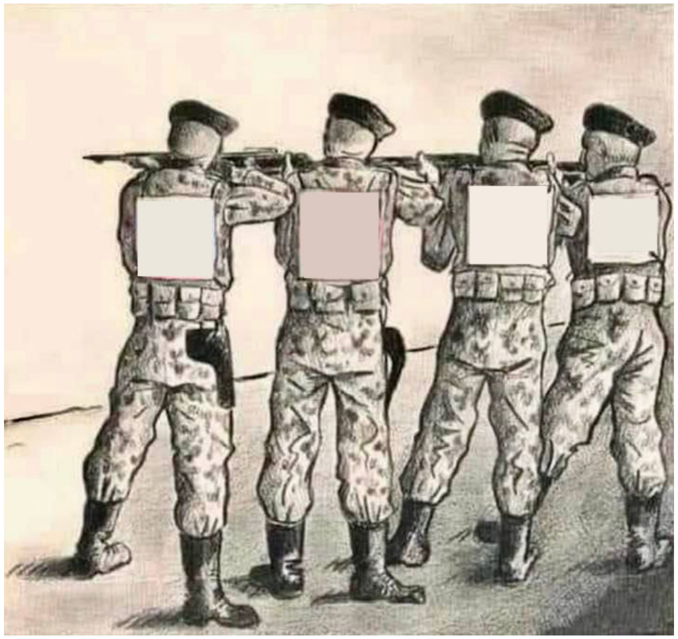
Can we use this "metalevel" approach to relate the truth of sentences described in a metalanguage to sentences describing those sentences?

Example: Can we use our truth predicate to formalize the truth or falsehood of people's statements?

$$\forall x.(says(john, x) \Rightarrow true(x))$$

Puzzle

You are taken prisoner by a drug cartel and told: *If you tell a lie, we will hang you. If you tell the truth, we will shoot you.* What do you say?



You say: *You will hang me.*
Result: They hang you *and* shoot you!
Suggestion: You should have asked if they meant *if and only if*.

Paradoxes

Unfortunately, trying to use a logic to define a truth predicate is problematic.

We run the risk of *paradoxes* (sentences that are both true and false / neither true nor false).

This sentence is false.

Also *nonsense terms* (terms that do not refer to anything).

The set of all sets that do not contain themselves

Results

We *can* completely formalize Propositional Logic in Term Logic.

(1) We can formalize *some details* of Term Logic in Term Logic but not everything. (2) We can formalize *more* of Term Logic in FOL, but we end up with *nonstandard models*. (3) We can eliminate nonstandard models using SOL, but it is complicated and there is no complete proof procedure.

We can axiomatize a metalevel truth predicate; but, unless we are very, very careful, this can lead to unpleasant complications, e.g. paradoxes.