Introduction to Logic
Term Logic

Michael Genesereth
Computer Science Department
Stanford University

Motivation

Finite Worlds

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

Countably Infinite 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
- Fitch Proofs

Next Time
- Induction

Next Week
- Logic in Logic
- Equality

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*).

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

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 term language is the set of all *ground terms* that can be formed from the vocabulary of the language.

The *Herbrand base* for a term 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*.

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

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

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):

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

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

$same(0,0)$	$\neg succ(0,0)$	$\neg less(0,0)$
$\neg same(0,s(0))$	$succ(0,s(0))$	$less(0,s(0))$
$\neg same(0,s(s(0)))$	$\neg succ(0,s(s(0)))$	$less(0,s(s(0)))$
…	…	…

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:

$same(0,0)$

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

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

…

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:

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

Successor

Positives:

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

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:

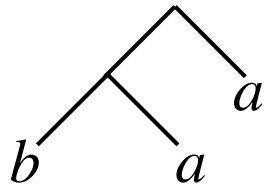
$$\forall x.\forall y.\forall z.(less(x,y) \wedge less(y,z) \Rightarrow less(x,z))$$

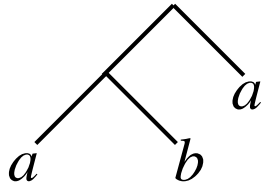
Irreflexivity:

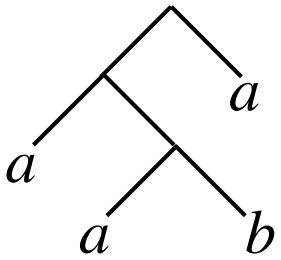
$$\forall x.\neg less(x,x)$$

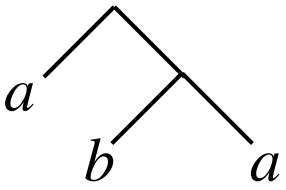
Example - Trees

Trees





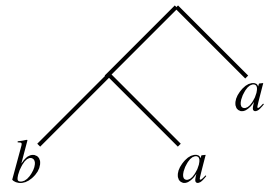




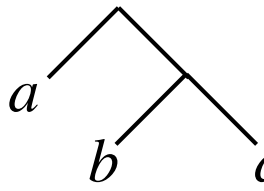
Tree Vocabulary

Object constants: a, b

Binary function constants: $cons$



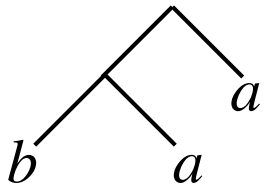
$cons(cons(b,a),a)$



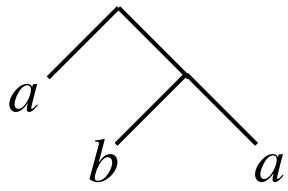
$cons(a,cons(b,a))$

Tree Vocabulary

Object constants: a, b
Binary function constants: *cons*



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



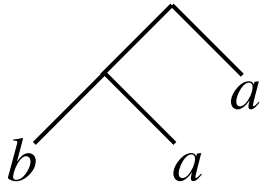
$cons(a, cons(b, a))$

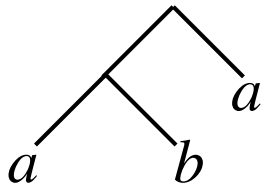
Unary relation constants: *symmetric*, *uniform*, ...
Binary relation constant: *subtree*, *congruent*, *mirror*, ...

Congruence

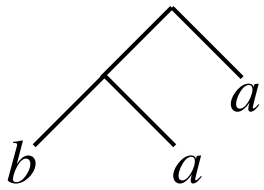
Two trees are *congruent* if and only if they have the *same shape*. (Labels on leaf nodes irrelevant.)

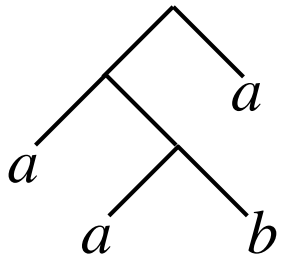
Examples:

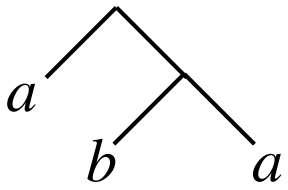




Non-Examples:







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

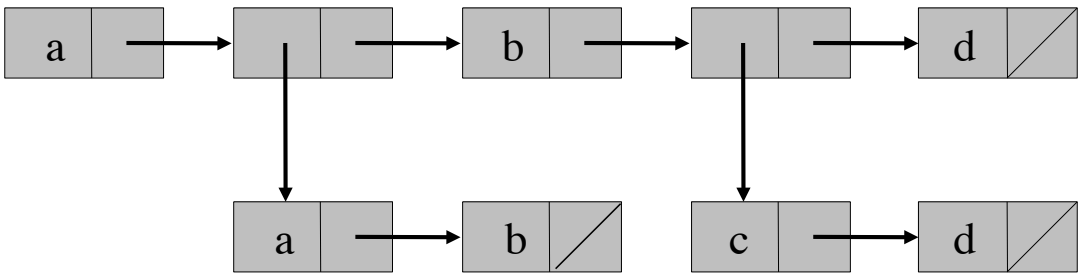
Flat Lists:

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

Nested Lists:

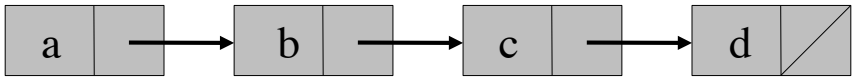
$$[a, [a, b], b, [c, d], d]$$

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

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

Definition :

$$\forall y.append(nil,y,y)$$
$$\forall x.\forall y.\forall z.\forall w.(append(y,z,w) \Rightarrow append(cons(x,y),z,cons(x,w)))$$

What else do we need?

Fitch Proofs

Propositional Logic

$$\{m \Rightarrow p \vee q, p \Rightarrow q\} \models m \Rightarrow q?$$

m	p	q	$m \Rightarrow p \vee q$	$p \Rightarrow q$	$m \Rightarrow q$
1	1	1	1	1	1
1	1	0	1	0	0
1	0	1	1	1	1
1	0	0	0	1	0
0	1	1	1	1	1
0	1	0	1	0	1
0	0	1	1	1	1
0	0	0	1	1	1

Relational Logic

$$\{p(a) \vee p(b), \forall x.(p(x) \Rightarrow q(x))\} \models \exists x.q(x)?$$

$p(a)$	$p(b)$	$q(a)$	$q(b)$	$p(a) \vee p(b)$	$\forall x.(p(x) \Rightarrow q(x))$	$\exists x.q(x)$
1	1	1	1	1	1	1
1	1	1	0	1	0	1
1	1	0	1	1	0	1
1	1	0	0	1	0	0
1	0	1	1	1	1	1
1	0	1	0	1	1	1
1	0	0	1	1	0	1
1	0	0	0	1	0	0
0	1	1	1	1	1	1
0	1	1	0	1	0	1
0	1	0	1	1	1	1
0	1	0	0	1	0	0
0	0	1	1	0	1	1
0	0	1	0	0	1	1
0	0	0	1	0	1	1
0	0	0	0	0	1	0

Functional Logic

Object constants: *n*
Unary relation constants: *k*
Factoids in Herbrand Base: *infinite*
Interpretations: *infinite*

Logical Rules of Inference

Negation Introduction
Negation Elimination

And Introduction
And Elimination

Or Introduction
Or Elimination

Assumption
Implication Elimination

Biconditional Introduction
Biconditional Elimination

Rules of Inference for Quantifiers

Universal Introduction
Universal Elimination

Existential Introduction
Existential Elimination

Domain Closure
Induction (Linear, Tree, Structural) *New!!*

Example - Spin

Problem

On transition, a particle goes from spin zero to spin positive or spin negative and then goes to spin zero on second transition. Show that if it has spin zero, it will have spin zero after two transitions.

Givens:

$\forall x{:}(zero(x) \Rightarrow pos(f(x)) \mid neg(f(x)))$

$\forall x{:}(pos(x) \Rightarrow zero(f(x)))$

$\forall x{:}(neg(x) \Rightarrow zero(f(x)))$

Goal:

$\forall x{:}(zero(x) \Rightarrow zero(f(f(x))))$

Proof

1. $\forall x.(zero(x) \Rightarrow pos(f(x)) \vee neg(f(x)))$ Premise
2. $\forall x.(pos(x) \Rightarrow zero(f(x)))$ Premise
3. $\forall x.(neg(x) \Rightarrow zero(f(x)))$ Premise

Proof

1.	$\forall x.(zero(x) \Rightarrow pos(f(x)) \vee neg(f(x)))$	Premise
2.	$\forall x.(pos(x) \Rightarrow zero(f(x)))$	Premise
3.	$\forall x.(neg(x) \Rightarrow zero(f(x)))$	Premise
4.	$\mid zero(c)$	Assumption

Proof

1.	$\forall x.(zero(x) \Rightarrow pos(f(x)) \vee neg(f(x)))$	Premise
2.	$\forall x.(pos(x) \Rightarrow zero(f(x)))$	Premise
3.	$\forall x.(neg(x) \Rightarrow zero(f(x)))$	Premise
4.	$\vert\ zero(c)$	Assumption
5.	$\vert\ zero(c) \Rightarrow pos(f(c)) \vee neg(f(c))$	UE: 1

Proof

1.	$\forall x.(zero(x) \Rightarrow pos(f(x)) \vee neg(f(x)))$	Premise
2.	$\forall x.(pos(x) \Rightarrow zero(f(x)))$	Premise
3.	$\forall x.(neg(x) \Rightarrow zero(f(x)))$	Premise
4.	$zero(c)$	Assumption
5.	$zero(c) \Rightarrow pos(f(c)) \vee neg(f(c))$	UE: 1
6.	$pos(f(c)) \vee neg(f(c))$	IE: 5, 4

Proof

1.	$\forall x.(zero(x) \Rightarrow pos(f(x)) \vee neg(f(x)))$	Premise
2.	$\forall x.(pos(x) \Rightarrow zero(f(x)))$	Premise
3.	$\forall x.(neg(x) \Rightarrow zero(f(x)))$	Premise
4.	$zero(c)$	Assumption
5.	$zero(c) \Rightarrow pos(f(c)) \vee neg(f(c))$	UE: 1
6.	$pos(f(c)) \vee neg(f(c))$	IE: 5, 4
7.	$pos(f(c)) \Rightarrow zero(f(f(c)))$	UE: 2
8.	$neg(f(c)) \Rightarrow zero(f(f(c)))$	UE: 3

Proof

1.	$\forall x.(zero(x) \Rightarrow pos(f(x)) \vee neg(f(x)))$	Premise
2.	$\forall x.(pos(x) \Rightarrow zero(f(x)))$	Premise
3.	$\forall x.(neg(x) \Rightarrow zero(f(x)))$	Premise
4.	$zero(c)$	Assumption
5.	$zero(c) \Rightarrow pos(f(c)) \vee neg(f(c))$	UE: 1
6.	$pos(f(c)) \vee neg(f(c))$	IE: 5, 4
7.	$pos(f(c)) \Rightarrow zero(f(f(c)))$	UE: 2
8.	$neg(f(c)) \Rightarrow zero(f(f(c)))$	UE: 3
9.	$zero(f(f(c)))$	OE: 6, 7, 8

Proof

1.	$\forall x.(zero(x) \Rightarrow pos(f(x)) \vee neg(f(x)))$	Premise
2.	$\forall x.(pos(x) \Rightarrow zero(f(x)))$	Premise
3.	$\forall x.(neg(x) \Rightarrow zero(f(x)))$	Premise
4.	$zero(c)$	Assumption
5.	$zero(c) \Rightarrow pos(f(c)) \vee neg(f(c))$	UE: 1
6.	$pos(f(c)) \vee neg(f(c))$	IE: 5, 4
7.	$pos(f(c)) \Rightarrow zero(f(f(c)))$	UE: 2
8.	$neg(f(c)) \Rightarrow zero(f(f(c)))$	UE: 3
9.	$zero(f(f(c)))$	OE: 6, 7, 8
10.	$zero(c) \Rightarrow zero(f(f(c))))$	II: 4, 9

Proof

1.	$\forall x.(zero(x) \Rightarrow pos(f(x)) \vee neg(f(x)))$	Premise
2.	$\forall x.(pos(x) \Rightarrow zero(f(x)))$	Premise
3.	$\forall x.(neg(x) \Rightarrow zero(f(x)))$	Premise
4.	$zero(c)$	Assumption
5.	$zero(c) \Rightarrow pos(f(c)) \vee neg(f(c))$	UE: 1
6.	$pos(f(c)) \vee neg(f(c))$	IE: 5, 4
7.	$pos(f(c)) \Rightarrow zero(f(f(c)))$	UE: 2
8.	$neg(f(c)) \Rightarrow zero(f(f(c)))$	UE: 3
9.	$zero(f(f(c)))$	OE: 6, 7, 8
10.	$zero(c) \Rightarrow zero(f(f(c))))$	II: 4, 9
11.	$\forall x.(zero(x) \Rightarrow zero(f(f(x))))$	UI: 10

Analysis

Logical Entailment and Provability

A set of premises Δ *logically entails* a conclusion φ ($\Delta \models \varphi$) if and only if every truth assignment that satisfies Δ also satisfies φ.

If there exists a *finite* proof of a sentence ϕ from a set Δ of premises using the rules of inference in R, we say that ϕ is *provable* from Δ using R (written $\Delta \vdash_R \phi$).

Soundness and Completeness

A proof system is *sound* if and only if every provable conclusion is logically entailed.

$$\text{If } \Delta \vdash \phi, \text{ then } \Delta \models \phi.$$

A proof system is *complete* if and only if every logical conclusion is provable.

$$\text{If } \Delta \models \phi, \text{ then } \Delta \vdash \phi.$$

Fitch Proof System

Theorem: Fitch is sound and complete for **Relational Logic**.

$$\Delta \models \phi \text{ if and only if } \Delta \vdash_{\text{Fitch}} \phi.$$

Theorem: Fitch is *sound* for **Term Logic** but it is *not complete*.

$$\text{if } \Delta \vdash_{\text{Fitch}} \phi, \text{ then } \Delta \models \phi.$$

Bad News

Theorem: Satisfiability / logical entailment for TL are not decidable.

Theorem: There is *no* sound and complete proof procedure for Term Logic.

Diophantine Equations or halting problem

Diophantine equation:

$$3x^2 = 1$$

Diophantine equation in Peano Arithmetic:

$$\exists x.\exists y.(\mathit{times}(x,x,y) \wedge \mathit{times}(s(s(s(0))),y,s(0)))$$

Solvability Question:

Axioms of Arithmetic $\cup$
$\{\exists x.\exists y.(\mathit{times}(x,x,y) \wedge \mathit{times}(s(s(s(0))),y,s(0)))\}$

It is known that the question of solvability of Diophantine equations is *not* decidable.

Two Types of Completeness

A *proof procedure* is complete if and only if everything that is logically entailed is provable.

A *set of sentences* Δ is complete if and only if for every sentence ϕ, either $\Delta \models \phi$ or $\Delta \models \neg \phi$.

Explanation

Given a finite set of sentences and a complete proof procedure, logical entailment is semidecidable.

If a set of sentences is complete, then logical entailment is decidable. (We try to prove the sentence and its negation, and one of the two will terminate if complete.)

Our axiomatization of Peano Arithmetic in Term Logic is finite and complete. **Assume** we had a sound and complete proof procedure. Then we would be able to decide the satisfiability of any Diophantine equation. **Contradiction**.
Upshot: there is no sound and complete proof procedure for Term Logic.

Compactness

A logic is *compact* if and only if every unsatisfiable set of sentences has a finite subset that is unsatisfiable.

Theorem: Propositional Logic is compact.
Theorem: Relational Logic is compact.

But...

A logic is *compact* if and only if every unsatisfiable set of sentences has a finite subset that is unsatisfiable.

Theorem: Propositional Logic is compact.
Theorem: Relational Logic is compact.
Theorem: Term Logic is *not* compact.

$$\{p(0), p(s(0)), p(s(s(0))), \ldots, \exists x.\neg p(x)\}$$

Significance: *Some conclusions in Term Logic have only infinite proofs*.

First-Order Logic (FOL)

First-Order Logic is logic with the same syntax as Term Logic but with a different semantics.

FOL is compact.
FOL has a sound and complete proof procedure!
FOL is (almost exclusively) taught in courses on Logic.

But...

There are things we can express in Term Logic that cannot be expressed in FOL, e.g. Peano Arithmetic, transitive closure.

Even though there is no complete proof procedure, Term Logic is ***not*** inferentially weaker than First-Order Logic. In fact, it is *stronger* than First-Order Logic.

If we can prove a conclusion in FOL, we can prove the result in Term Logic. In fact, using induction, we can prove *more* things in Term Logic than in FOL.

The issue is that there are *more* things that are true. We just cannot prove them all.