Intro to Predicate Logic

Sipser Ch. 6.2; Papadimitriou Ch. 5

Agenda:

- What predicate logic is. And some variants...
- What are the associated computational problems?
- Some results on complexity/decidability of those problems (indicates expressive power of predicate logic)
- Probably I don't have time for all the following material...

Intro to Predicate Logic

Logical inference:

1. Every student is honest
2. Harry is a student
3. deduce that...

Statements about the world (maybe, *a* world) where you can automatically deduce stuff

Propositional logic doesn't have the expressive power to capture these statements.
Next: define (first order) *predicate logic*; study the associated computational problems: decidable? In **P**, **NP**?

From a CS perspective: look for more powerful knowledge representation language that can describe situations with no fixed number of individuals.

Propositional logic worlds

Vocabulary of propositional logic refers to some fixed number of facts $\rightarrow$ fixing a vocabulary of propositions $p_1 \dots p_n$ restricts us to "state of the world" description using exactly n bits.
Cannot model statements about unspecified numbers of individuals.

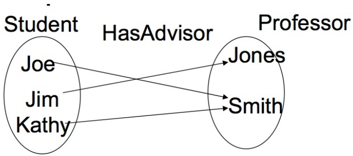
Given a collection of propositional logic statements we can identify their *signature:* set of propositions p, q, r, s —
Possible world (a.k.a. truth valuation/assignment) e.g.:

- p, q, r=TRUE; s=FALSE
- p, q TRUE; r, s FALSE

Predicate logic worlds (example)

Vocabulary (a.k.a. *signature* or *language*): set of predicates, functions, constants
Possible world (a.k.a. **interpretation**; *model*; *structure*): includes *domain*, the set of values that variables can take, includes the constants

- `Student(x), Professor(y), HasAdvisor(x,y)`

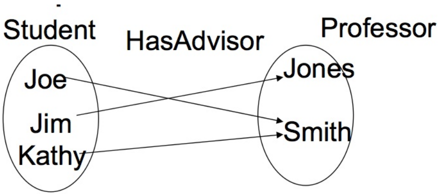


Predicate Logic Worlds

Vocabulary: set of predicates, functions, constants

`Student(x), Professor(y), HasAdvisor(x,y)`

interpretation (a possible world)



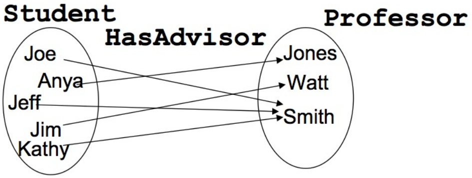
Represented formally, this interpretation looks like:

Domain = {Joe, Jim, Kathy, Jones, Smith}
Student = {Joe, Jim, Kathy}
Professor ={Jones, Smith}
HasAdvisor = {(Joe,Smith), (Jim, Jones), (Kathy, Smith)}

Vocabulary: set of predicates, functions, constants

`Student(x), Professor(y), HasAdvisor(x,y)`

Possible world



For the same vocabulary, have infinitely many possible worlds!

Predicate Logic, formally

Vocabulary: collections of

- constants – say $\{c_i : i \geq 0\}$.
  Constants are names for individuals. E.g.: 0, 1
- function symbols – say $\{f_i : i \geq 0\}$.
  May be of different number of arguments (arities) E.g.: $+(x, y)$
- predicate symbols – say $\{p_i : i \geq 0\}$.
  each with its own number of arguments (arity)

Collections don't have to be finite: a vocabulary can be infinite

Predicate Logic, formally

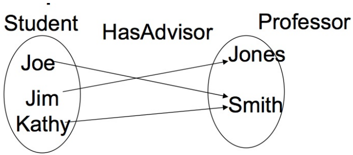
Worlds: For a Predicate Logic vocabulary V, an interpretation for V consists of:

- A set D (the domain or universe)
- For every k-ary relation symbol R in V, a k-ary relation on D
- For every k-ary function symbol f in V, a k-ary function on D
- For every constant symbol c in V, an element of D

Some books call this a **model** for V, or a **structure** for V

Example Interpretation

Student is 1-ary, **Professor** is 1-ary, **HasAdvisor** is 2-ary; constants could also be thought of as 0-ary functions



Vocabulary:

- No functions
- Predicates: `Student(x)`, `Professor(y)`, `HasAdvisor(x,y)`
- No constants

Domain = {Joe, Jim, Kathy, Jones, Smith}
Student(x) true for $x \in$ {Joe, Jim, Kathy}
Professor(y) true for $y \in$ {Jones, Smith}
HasAdvisor(x,y) true for $(x, y) \in$ {(Joe,Smith), (Jim, Jones), (Kathy, Smith)}

Predicate Logic, formally (more examples)

reminder: For vocab V, interpretation for V comprises:

- A set D (the domain or universe)
- For every k-ary relation symbol R in V, a k-ary relation on D
- For every k-ary function symbol f in V, a k-ary function on D
- For every constant symbol c in V, an element of D

Example: $V_{field} :=$ 2-ary functions $+$, $*$, constants: 0, 1; for ordered field, 2-ary predicate $<$

One interpretation: Domain = Integers

$+$,*,$<$ usual arithmetic operators and comparison 0,1,$=$ as usual

Predicate Logic, formally (more examples)

reminder: For vocab V, interpretation for V comprises:

- A set D (the domain or universe)
- For every k-ary relation symbol R in V, a k-ary relation on D
- For every k-ary function symbol f in V, a k-ary function on D
- For every constant symbol c in V, an element of D

Example: $V_{field} :=$ 2-ary functions $+$, $*$, constants: 0, 1; for ordered field, 2-ary predicate $<$

One interpretation: Domain = Integers

$+$,*,$<$ usual arithmetic operators and comparison 0,1,= as usual

Alternatively: could have domain = Real numbers

Predicate Logic, formally (more examples)

reminder: For vocab V, interpretation for V comprises:

- A set D (the domain or universe)
- For every k-ary relation symbol R in V, a k-ary relation on D
- For every k-ary function symbol f in V, a k-ary function on D
- For every constant symbol c in V, an element of D

Example: $V_{field} :=$ 2-ary functions $+$, $*$, constants: 0, 1; for ordered field, 2-ary predicate $<$

One interpretation: Domain = Integers

$+$,*,$<$ usual arithmetic operators and comparison 0,1,$=$ as usual

Alternatively: could have domain $=$ Real numbers

Alternatively: could have domain $=$ Real numbers; 0 $=$ the number 15; 1 $=$ the number 7

Notation

Suppose V is a vocabulary with n-ary predicate P and m-ary function symbol F.
If M is an interpretation for V, then M consists of

- domain of M (a set) denoted Dom(M) or $\|\mathrm{M}\|$
- interpretation for P, denoted P^M, an n-ary relation on M
- interpretation for F, denoted F^M, an m-ary function on M

Predicate Logic Statements

Formulae are statements about one or more objects in a world

"x is an honest student"

Sentences are statements about a world

Every student is honest

Some student is honest

Given an interpretation, a sentence should get the value TRUE or FALSE

Some computational challenges

Decision problems:

1. Given a sentence, it is TRUE in all interpretations? (then, said to be *valid*)
2. Is some given sentence satisfiable by some interpretation? (does it have a *model*?)
3. Given a sentence and an interpretation, is it true?

Item (3) raises question of how *infinite* interpretation is presented to an algorithm (a finite one can be presented as a list of domain elements). In fact, consider certain specific interpretations; e.g. "first-order theory of real arithmetic": sentences where variables range over $\mathbb{R}$, standard operators $+$, $\times$, $\exists$, $\forall$

Predicate Logic(s)

Many variants (it's a rich area!) Expressions may use quantifiers $\forall$, $\exists$ (you know what those are, right?)

- e.g. restriction to "existential theories", limit to statements where there's just one quantifier, $\exists$ at start of statement.
- **QBF:** "quantified boolean formulae" — propositional logic with quantifiers, **PSPACE**-complete to determine whether a given formula is true/satisfiable.
- First-Order Logic – quantifications over domain only: "$\forall$ x", "$\exists$ x": x in domain
- There are other logics, e.g. richer than first-order. Second-order logic, FixedPoint Logic, Logic with Counting Quantifiers etc.
  - More on these in other courses (e.g. Logic Automata Games, Theory of Data and Knowledge Bases)

Terms

Semantically, a *term* should represent an element of the domain.
Syntactically, recursive definition:

- Every constant of vocabulary V is a term. So is every variable.
- If f_i is an n-ary function symbol of V and $t_1, \ldots, t_n$ are terms, then $f_i(t_1, \ldots, t_n)$ is a term.

Examples

- $f(x, g(2, y))$ is a term, where f, g are function symbols and x, y are variables.
- $+(x, *(3, y))$ is a term in the vocab for arithmetic; usually written as $x + (3 * y)$

Recursive Definition of Formulae

(Semantically, something that evaluates to true or false. Value may depend on values of "free variables" in the formula, e.g. formula "$x = y$".)

- If p_i is an n-ary predicate symbol in V and $t_1, \ldots, t_n$ are terms of V, then:
  - $p_i(t_1, \ldots, t_n)$ is an atomic formula
  - $t_i = t_j$ is an atomic formula
- If A and B are formulas, then so are:
  - $A \wedge B$, T, F, $A \vee B$, $\neg A$, $A \rightarrow B$ (could also include $A \leftrightarrow B$, $A \oplus B$, other propositional connectives...)
  - $\forall x_i A$, $\exists x_i A$, where x_i is a variable (usually, x_i appears in A).

$$\forall x \text{Student}(x) \Rightarrow \exists y(\text{HasAdvisor}(x, y) \land \text{Professor}(y))$$

Informally: "Every student has an advisor that is a professor."

True in the example interpretation

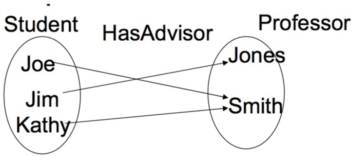
$$\exists x(\text{Student}(x) \land \text{Professor}(x))$$

"There is a student who is also a professor"

$$\text{Student}(x) \land \exists y[\text{Student}(y) \land \neg(x = y) \land$$
$$\exists z(\text{HasAdvisor}(x, z) \land \text{HasAdvisor}(y, z))]$$

"x is a student and there is some other student who has the same advisor as x"

True of Joe and Kathy



Free and bound Variables

In $\forall x \; A(x, y)$, the variable x is said to be bound; y is free.
Generally, there is a recursive definition of the free variables of a formula.

- x occurs free in any $A(t_1 \ldots t_n)$ where some t_i contains x
- x occurs free in $t_1 = t_2$, where t_1 or t_2 contains x
- x occurs free in:
  - $\forall y \; A$ or $\exists y \; A$ if x occurs free in A and x is not y
  - $\neg A$ if x occurs free in A.
  - $A \wedge B$, $A \vee B$, $A \Rightarrow B$, ... if x occurs free in either A or B

If x occurs in a formula ϕ, and x is not free in ϕ, then x is a *bound* variable of ϕ

Write $\phi(x_1, \ldots, x_n)$ if $x_1, \ldots, x_n$ are all the free variables of ϕ.
A sentence is a formula with no free variables.

Examples

$$\forall x[\text{Student}(x) \Rightarrow \text{HasAdvisor}(x, y)]$$

x is bound, y is free

$$\text{Student}(x) \wedge \exists z(\text{Professor}(z) \wedge \text{Knows}(x, z) \wedge$$

$$(\forall x[\text{Student}(x) \Rightarrow \text{HasAdvisor}(x, y)])$$

Examples

$$\forall x[\text{Student}(x) \Rightarrow \text{HasAdvisor}(x, y)]$$

x is bound, y is free

$$\text{Student}(x) \wedge \exists z(\text{Professor}(z) \wedge \text{Knows}(x, z)\wedge$$

$$(\forall x[\text{Student}(x) \Rightarrow \text{HasAdvisor}(x, y)])$$

y is free and x is free!

$$\forall x[\text{Student}(x) \Rightarrow \exists z(\text{Professor}(z) \wedge \text{Knows}(x, z)\wedge$$

$$(\forall x[\text{Student}(x) \Rightarrow \text{HasAdvisor}(x, y)])]$$

Examples

$$\forall x[\text{Student}(x) \Rightarrow \text{HasAdvisor}(x, y)]$$

x is bound, y is free

$$\text{Student}(x) \wedge \exists z(\text{Professor}(z) \wedge \text{Knows}(x, z) \wedge$$

$$(\forall x[\text{Student}(x) \Rightarrow \text{HasAdvisor}(x, y)])$$

y is free and x is free!

$$\forall x[\text{Student}(x) \Rightarrow \exists z(\text{Professor}(z) \wedge \text{Knows}(x, z) \wedge$$

$$(\forall x[\text{Student}(x) \Rightarrow \text{HasAdvisor}(x, y)]))]$$

only y is free

$$\exists y \forall x[\text{Student}(x) \Rightarrow \text{HasAdvisor}(x, y)]$$

Nothing free – a sentence.

Semantics of First Order Logic

What it means formally for a sentence ϕ to hold in interpretation M, as in "bottom up" semantics of propositional logic.

For ϕ with free variables, can only say whether it is true relative to some valuation = assignment of each variable to an element of Dom(M) (also called a variable binding or variable assignment)

Define $\mathrm{M}, v \models \phi$

where M is an interpretation, v a valuation for free variables of ϕ

"M, v satisfies ϕ"
"M, v models ϕ"
"M, v entails ϕ"

E.g. if $\phi(x)$ is "x is a student who shares an advisor" (from prior slide), M is the model from before, then

$\mathrm{M}, \{x \to \mathrm{Kathy}\} \models \phi(x)$
$\mathrm{M}, \{x \to \mathrm{Joe}\} \models \phi(x)$

Semantics of FO Logic: Base Cases

Let M be an interpretation and v a valuation for free variables in formula ϕ. We define $\mathrm{M}, v \models \phi$ as follows.

$$\mathrm{M}, v \models t_i = t_j \text{ iff } v[t_i] = v[t_j]$$

where $v[t_i]$ is "the extension of v to term t_i" defined inductively
$v[x_i] = v(x_i)$, $v[c] = c^{\mathrm{M}}$, x_i a variable c a constant
$v[F(t_1 \ldots t_n)] = F^{\mathrm{M}}(v[t_1] \ldots v[t_n])$

Example: V_{Field} from before

interpretation M: domain=integers, $*$ usual multiplication, $+$ is usual addition...

v valuation taking: x to 4, y to 4, z to 2
Consider terms $t_1 = x + y$, $t_2 = y * z$ then $v[t_1] = 8$ $v[t_2] = 8$
so $\mathrm{M}, v \models t_1 = t_2$

Semantics of FO Logic: Base Cases

Let M be an interpretation and v a valuation for free variables in formula ϕ. We define $\mathsf{M}, v \models \phi$ as follows.

$$\mathsf{M}, v \models P(t_1, \ldots t_n) \text{ iff } v[t_i], \ldots v[t_n] \in P^{\mathsf{M}}$$

where $v[t_i]$ is defined inductively on previous slide

Example: V_{Field} from before

- M the integer interpretation (domain=integers, $*$ is usual multiplication, $+$ is usual addition, $<$ usual inequality...)
- v valuation taking: x to 4, y to 4, z to 2

Then:

$$\mathsf{M}, v \models z + x < y + x$$

It is not true that $\mathsf{M}, v \models x < y$ (written $\mathsf{M}, v \not\models x < y$)

Semantics of FO Logic: Connectives and quantifiers

Let M be an interpretation and v a valuation for free variables in formula ϕ. We define $\mathrm{M}, v \models \phi$ as follows.

$$M, v \models A \wedge B \text{ iff } M, v \models A \text{ and } M, v \models B$$

$$M, v \models A \vee B \text{ iff } M, v \models A \text{ or } M, v \models B$$

$$M, v \models \neg A \text{ iff it is not the case that } M, v \models A$$

Other connectives can be defined using these.

$$M, v \models \exists x\ \phi \text{ iff}$$

there is some element d in $Dom(\mathrm{M})$ such that:
$\mathrm{M}, v + (x \mapsto d) \models \phi$
$v + (x \mapsto d)$ = function that extends v to map x to d (overwriting any other assignment to x if need be)

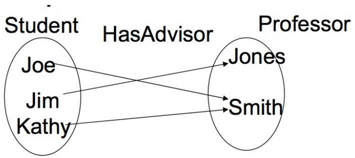
Let M be an interpretation and v a valuation for free variables in formula ϕ. We define $\mathrm{M}, v \models \phi$ as follows.

$$M, v \models \forall x\ \phi \text{ iff}$$

for *every* element d in $Dom(\mathrm{M})$ such that:
if $v + (x \mapsto d)$ = function that extends/overwrites v to map x to d then $\mathrm{M}, v + (x \mapsto d) \models \phi$

Example of checking a finite model

M=



$\phi = \exists x(\text{Student}(x) \wedge \exists y(\text{Professor}(y) \wedge \text{HasAdvisor}(x, y)))$

$\mathsf{M} \models \phi$ means $\mathsf{M}, \{\} \models \phi$ where $\{\}$=empty valuation

Check: does there exist element d of $\mathit{Domain}(\mathsf{M})$, with

$\mathsf{M}, \{\} + (x \mapsto d) \models \text{Student}(x) \wedge \exists y(\text{Professor}(y) \wedge \text{HasAdvisor}(x, y))$

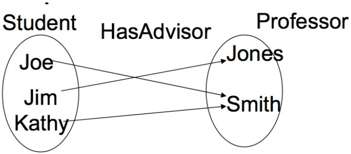
Try $d = \text{Joe}$; Check

$\mathsf{M}, \{\} + (x \mapsto \text{Joe}) \models \text{Student}(x) \wedge \exists y(\text{Professor}(y) \wedge \text{HasAdvisor}(x, y))$

Check $\mathsf{M}, \{\} + (x \mapsto \text{Joe}) \models \text{Student}(x)$ $\quad$ $\text{Joe} \in \text{Student}^{\mathsf{M}}$ $\rightarrow$ OK

Check $\mathsf{M}, \{\} + (x \mapsto \text{Joe}) \models \exists y(\text{Professor}(y) \wedge \text{HasAdvisor}(x, y))$

Example (continued)

$\mathsf{M}=$ 

$$\phi = \exists x(\text{Student}(x) \wedge \exists y(\text{Professor}(y) \wedge \text{HasAdvisor}(x, y)))$$

Try d = Joe; Check
$\mathsf{M}, \{\} + (x \mapsto \text{Joe}) \models \text{Student}(x) \wedge \exists y(\text{Professor}(y) \wedge \text{HasAdvisor}(x, y))$

Check $\mathsf{M}, \{\} + (x \mapsto \text{Joe}) \models \text{Student}(x)$ Joe $\in \text{Student}^{\mathsf{M}} \rightarrow$ OK

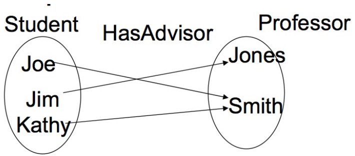
$\rightarrow$ Check $\mathsf{M}, \{\} + (x \mapsto \text{Joe}) \models \exists y(\text{Professor}(y) \wedge \text{HasAdvisor}(x, y))$

Sub Check: does there exist element of d of *Domain*(M), with
$\mathsf{M}, \{\} + (x \mapsto \text{Joe}) + (y \mapsto d) \models \text{Professor}(y) \wedge \text{HasAdvisor}(x, y)$

Try: d = Smith

Check: $\mathsf{M}, \{\} + (x \mapsto \text{Joe}) + (y \mapsto \text{Smith}) \models \text{Professor}(y) \wedge \text{HasAdvisor}(x, y)$

Example (continued)

$\mathsf{M}=$ 

$\phi = \exists x(\mathrm{Student}(x) \wedge \exists y(\mathrm{Professor}(y) \wedge \mathrm{HasAdvisor}(x,y)))$

Try: $d = \mathrm{Smith}$

$\rightarrow$ Check: $\mathrm{M}, \{\} + (x \mapsto \mathrm{Joe}) + (y \mapsto \mathrm{Smith}) \models \mathrm{Professor}(y) \wedge \mathrm{HasAdvisor}(x,y)$

SubCheck 1: $\mathrm{M}, \{\} + (x \mapsto \mathrm{Joe}) + (y \mapsto \mathrm{Smith}) \models \mathrm{Professor}(y)$

$\mathrm{Smith} \in \mathrm{Professor}^{\mathrm{M}} \rightarrow \mathrm{OK}$

SubCheck 2: $\mathrm{M}, \{\} + (x \mapsto \mathrm{Joe}) + (y \mapsto \mathrm{Smith}) \models \mathrm{HasAdvisor}(x,y)$

$(\mathrm{Joe}, \mathrm{Smith}) \in \mathrm{HasAdvisor}^{\mathrm{M}} \rightarrow \mathrm{OK}$

Notes

- Previous example is a proper recursive algorithm that checks for satisfaction, given a finite model.
- In each case I hit an existential quantifier, I had to choose a d: in the example I just showed the correct guess. In general, the algorithm would have to recursively try every possible d $\rightarrow$ if no d works, returns failure.

The Logic-Computation Connection

We have a defined meaning for

$$M, v \models \phi$$

where M is an interpretation, v a valuation for free variables of ϕ.
First basic computational problem related to Predicate Logic:
Given a finite model M and a first-order sentence ϕ, does $M \models \phi$?

Model Checking Problem for First-Order Logic:

$\{\langle M, \phi \rangle \ : \ M \models \phi\}$
Decidable $\rightarrow$ Use previous algorithm
Time complexity: $|M|^{|\phi|}$
Can do much better for special kinds of sentences that arise frequently.

Model Checking

$$\text{MCFO} = \{\langle \text{M}, \phi\rangle \ : \ \text{M} \models \phi\}$$

Also can say: MCFO$\in$ **PSPACE**, MCFO is **NP**-hard

NP-hardness: By reduction from SAT

In the reduction, always produce model for vocabulary with constants **True** and **False**, M_{bools} that consists just of two elements, one of which interprets the constant **True**, the other interpreting **False**.

Given propositional formula

$$\phi = (p_1 \vee \neg p_2 \vee p_3) \wedge \ldots$$

produce $(\text{M}_{bools}, \phi')$, where

$$\phi' = \exists x_1 \ldots x_n \Big[(x_1 = \text{TRUE} \vee \neg x_2 = \text{TRUE} \vee x_3 = \text{TRUE}) \wedge \ldots \Big]$$

Observations: (1) the reduction did not require much work; in a sense model-checking FOPL is easily seen to be "more complex" than SAT-solving.
(2) M does not depend on SAT-instance ϕ; ϕ is encoded in ϕ'

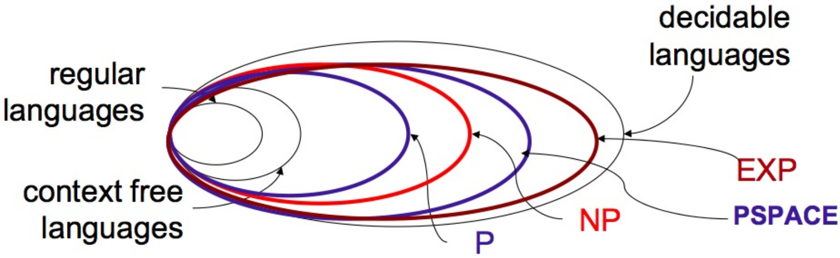
Model Checking FO Logic

$$\text{MCFO} = \{\langle \text{M}, \phi \rangle \ : \ \text{M} \models \phi\}$$

In **PSPACE**, **NP**-hard...

In fact, it is **PSPACE**-complete. For every language L decidable by a **PSPACE** machine, there is a PTIME reduction from L to MCFO.

Complexity Summary



PSPACE could be the same as PTIME (i.e. P)
PSPACE could be the same as **NP**
PSPACE could be the same as **EXP**
PSPACE could be different from all of them

Properties of Formulae

A sentence ϕ is satisfiable if there is some interpretation M such that $M \models \phi$

A sentence ϕ is valid (or, is a tautology) if for every model M, $M \models \phi$

A sentence ϕ is a contradiction if there is no M such that $M \models \phi$

Two sentences ϕ_1 and ϕ_2 are equivalent if they have the same models (same as $\phi_1 \leftrightarrow \phi_2$ is valid).

Examples

$\forall x(A(x) \vee \neg A(x))$ Valid

$A(c) \Rightarrow \exists y\ A(y)$ Valid

$(\forall x\ \text{Student}(x) \Rightarrow \text{Hardworking}(x)) \wedge \text{Student}(\textit{Harry}) \wedge \neg\text{Hardworking}(\textit{Harry})$ Contradiction

$\forall x(\text{Man}(x) \Rightarrow \text{Mortal}(x))$

$\forall z\ (z + z > z)$

Properties of Formulae II

A sentence ϕ is finitely satisfiable if there is some finite model M such that $\mathsf{M} \models \phi$

A sentence ϕ is finitely valid if for every finite model M, $\mathsf{M} \models \phi$.

Example

$\exists x\ (x = x)\ \wedge\ \forall x \forall y \forall z((x < y \wedge y < z) \Rightarrow x < z) \wedge$
$\forall x\ \neg(x < x)\ \wedge\ \forall x \exists y\ (x < y)$
is satisfiable but not finitely satisfiable.

Computational View

Second basic computational problem:
Given a sentence ϕ is it finitely satisfiable?
Finite satisfiability problem
Semi-decidable
Enumerate models, and check, using recursive algorithm.
Undecidable
(this is Trakhtenbrot's Theorem)
Corollary: Finite Validity problem is not CE

Next: overview of proof of Trakhtenbrot's theorem — reduce TM acceptance problem to search for finite model for ϕ.
Given TM M and word w, construct $\phi(M)$; $\phi(M)$ has finite model iff M halts and accepts w.

Reducing computation to logic

Given M and string w compute a propositional sentence that codes accepting runs of M on w.

$\longrightarrow$ Given M and string w compute a first-order pred logic sentence that codes accepting runs of M on w.

Want computable f such that: M accepts $w \Leftrightarrow f(M, w)$ finitely satisfiable

Many-1 reduction from Accept_{TM} to $\text{FiniteSat}_{First\ Order\ Logic}$

Reduction produces a sentence that describes a (code of an) accepting run of M on w.

As in Cook's Theorem, sentence will be a conjunction of many clauses.

The domain of any (satisfying) finite model will consist of elements that represent time steps, and TM tape squares.

Describe a run with a propositional world

Given NTM M with bound n^k and string w compute a propositional formula that describes a (code of) run of M on w

Have propositions $\text{HasSymbol}_{i,j}(a)$ and $\text{HasHead}_{i,j}(q)$ for every tape symbol a, state q and every $i, j \leq n^k$ in this matrix.

$\text{HasSymbol}_{i,j}(a)$ holds iff run at cell (i, j) has symbol a

$\text{HasHead}_{i,j}(q)$ holds iff the head is on j at step i of the and state is q

Tape space j

Time i

	1	2	$\cdots$	n^k
1	(q_0, w_1)	w_2	$\cdots$	
2	w_1'	(q_1, w_2)		
$\vdots$				
$\vdots$				
n^k				

This corresponds to a run where
$\text{HasSymbol}_{1,1}(w_1)$
$\text{HasHead}_{1,1}(q_0)$
$\text{HasSymbol}_{1,2}(w_2)$
$\text{HasSymbol}_{2,1}(w_1')$
$\text{HasSymbol}_{2,2}(w_2)$
$\text{HasHead}_{2,2}(q_1)$
...are true
(Others, e.g. $\text{HasHead}_{1,2}(q_0)$ are false)

Coding Big Runs by Sentences

Describe a run with a

predicate logic world

Given any deterministic TM M and string w compute an FO sentence that describes a (code of) run of M on w

Have predicates $\mathsf{Time}(x)$, $\mathsf{Space}(y)$, $\mathsf{LessThan}(x_1, x_2)$
Have predicates $\mathsf{HasSymbol}_a(t, s)$ and $\mathsf{HasHead}_q(t, s)$ for every symbol a, state q

$\mathsf{Time}(x)$ holds iff x is a number $\leq$ number of steps of machine

$\mathsf{Space}(y)$ holds iff y is a number $\leq$ amount of space used by machine

$\mathsf{LessThan}(x_1, x_2)$ holds iff x_1 and x_2 are both in Time or both in Space and x_1 comes before x_2

$\mathsf{HasSymbol}_a(t, s)$ holds iff run at time t position s has symbol a

$\mathsf{HasHead}_q(t, s)$ holds iff the head is on place s at step t of the run and state is q

Tape space j

		1	2	$\cdots$	s_{max}
Time	1	(q_0, w_1)	w_2	$\cdots$	
	2	w_1'	(q_1, w_2)		
	$\vdots$				
	$\vdots$				
	t_{final}				

This run corresponds to a world where

$\mathsf{Time} = \{1, \ldots, t_{final}\}$
$\mathsf{Space} = \{1, \ldots, s_{max}\}$
$\mathsf{LessThan}=$ usual $<$ on numbers
$\mathsf{HasSymbol}_{w_1} = \{(1, 1), \ldots\}$
$\mathsf{HasHead}_{q_0} = \{(1, 1), \ldots\}$
$\mathsf{HasSymbol}_{w_2} = \{(1, 2), \ldots\}$

Clauses for LessThan

$$\forall x, y \ \mathrm{LessThan}(x, y) \Rightarrow \neg \mathrm{LessThan}(y, x)$$

$$\forall x, y, z \ \mathrm{LessThan}(x, y) \wedge \mathrm{LessThan}(y, z) \Rightarrow \mathrm{LessThan}(x, z)$$

The above makes sure that LessThan is a linear order. Also add

$$\forall x, y \ \mathrm{LessThan}(x, y) \Rightarrow (\mathrm{Time}(x) \wedge \mathrm{Time}(y)) \vee (\mathrm{Space}(x) \wedge \mathrm{Space}(y))$$

The above makes sure that LessThan may compare domain elements belonging to relation Time, or alternatively Space.

Sanity Clauses: world "looks like a run"

- LessThan is a linear order, whose domain includes everything inside the unary predicates Time and Space
- Time is closed downward under LessThan and similarly for Space
  - i.e. $(\text{Time}(x) \wedge y < x) \Rightarrow \text{Time}(y)$ etc.
- HasHead and HasSymbol hold only of Time/Space pairs:
  - $\forall xy \text{HasHead}_q(x, y) \Rightarrow \text{Time}(x) \wedge \text{Space}(y)$
  - $\forall xy \text{HasSymbol}_a(x, y) \Rightarrow \text{Time}(x) \wedge \text{Space}(y)$
- Every cell has at most one symbol
  $\forall x \forall y \text{HasSymbol}_a(x, y) \Rightarrow \neg \text{HasSymbol}_b(x, y)$ for $b \neq a$
- At most one cell in each row has the head. For every q, q' have:
  $\forall x \forall y \text{HasHead}_q(x, y) \Rightarrow \neg \exists y' \text{HasHead}_{q'}(x, y') \wedge y' \neq y$

Macros

It's useful to define/axiomatise new predicates in terms of LessThan...

- We can define $\mathrm{IsFirstTime}(x)$ to mean "$\mathrm{Time}(x)$ and for no y with $\mathrm{Time}(y)$ do we have $\mathrm{LessThan}(y, x)$", i.e. x is time 1
- Similarly can write a formula $\mathrm{IsFirstSpace}(x)$
- Similarly, can write $\mathrm{IsTime}_n(x)$, $\mathrm{IsSpace}_n(x)$ for every fixed n.

We also use the formula:

$\mathrm{Successor}(x, y) \Leftrightarrow$
$\mathrm{LessThan}(x, y) \wedge \neg\exists z(\mathrm{LessThan}(x, z) \wedge \mathrm{LessThan}(z, y))$
i.e. "$y = x + 1$"

Sentence that says "M has an accepting run on w"

Initial Configuration Clause
First row contains Initial State:
$\forall t_1 \forall s_1 \ldots \forall s_n$ [$\mathrm{IsTime}_1(t_1) \wedge \mathrm{IsSpace}_1(s_1) \wedge \ldots \wedge \mathrm{IsSpace}_n(s_n)] \Rightarrow$
$[\mathrm{HasSymbol}_{w_1}(t_1, s_1) \wedge \ldots \wedge \mathrm{HasSymbol}_{w_n}(t_1, s_n)$
$\wedge \mathrm{HasHead}_{q_0}(t_1, s_1) \wedge \forall z[\mathrm{Space}(z) \wedge \mathrm{LessThan}(s_n, z) \Rightarrow$
$\mathrm{HasSymbol}_{\perp}(t_1, z)]]$

Tape space j

Time		1	2	$\cdots$	s_{max}
	1	(q_0, w_1)	w_2	$\cdots$	$\perp$
	2				
	$\vdots$				
	$\vdots$				
	t_{final}				

$\perp$ is a blank symbol

For every transition $(q, w_1) \rightarrow (q_1, w_2, R)$ in machine M we add a clause:

$\forall t \forall s \forall s' \forall t'$ $\text{HasHead}_q(t, s) \wedge \text{HasSymbol}_{w_1}(t, s) \wedge$
$\text{Successor}(t, t') \wedge \text{Successor}(s, s') \Rightarrow [\text{HasHead}_{q_1}(t', s') \wedge \ldots]$

Tape space

	1	$\cdots$	s	s'	$\cdots$	s_{max}
1						
t			(q, w_1)	w_3	$\cdots$	
t'			w_2	(q_1, w_3)	$\cdots$	
$\vdots$						
t_{final}						

For every transition $(q, w_1) \rightarrow (q_1, w_2, Stay)$ in machine M we add:

$$\forall t \forall s \forall t' \ [\mathrm{HasHead}_q(t, s) \wedge \mathrm{HasSymbol}_{w_1}(t, s) \wedge \mathrm{Successor}(t, t')] \Rightarrow [\mathrm{HasSymbol}_{w_2}(t', s) \wedge \mathrm{HasHead}_{q_1}(t', s)]$$

Tape space

$$
\begin{array}{r|cccccc}
 & 1 & \cdots & s & & \cdots & s_{max} \\
\hline
1 & & & & & & \\
 & & & & & & \\
t & & & (q, w_1) & w_3 & \cdots & \\
t' & & & (q_1, w_2) & w_3 & \cdots & \\
\vdots & & & & & & \\
t_{final} & & & & & &
\end{array}
$$

Hopefully, you get the idea

Add left-move axioms, far-from-head axioms, final-state axioms...

Then need to show:

- Function $f(M, w)$ is computable
- Yes maps to Yes:
  if M accepts w, take the accepting run r and turn it into a structure $\mathrm{code}(r)$ using the coding function. From properties of an accepting run, we see that $\mathrm{code}(r)$ satisfies $f(W, w)$.
- No maps to No:
  Suppose $f(M, w)$ does have a (finite) model W. Interpret members of domain that satisfy Time as time steps, etc; reconstruct computation of M.

More Model Checking Problems

OK, so we know there's a semi-decision procedure for finite satisfiability (given FOPL sentence), but it's undecidable.

Third basic computational problem:

Given a sentence ϕ is it satisfiable? Is it valid?

Not immediately clear if either of these is semi-decidable

- If there is an infinite model, how would you find it?
- If there are no infinite models, how would you know this?

A Proof System (for FOL without equality)

We say formula $\phi(x_1, \ldots, x_n)$ is *valid* if $\forall x_1 \ldots \forall x_n \ \phi(x_1 \ldots x_n)$ holds in every model.

AS1 $A \Rightarrow (B \Rightarrow A)$

AS2 $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$

AS3 $(\neg A \Rightarrow \neg B) \Rightarrow (B \Rightarrow A)$

AS4 $(\forall x A) \Rightarrow A(t/x)$ where t is any term in which x is not free and t/x means substitute t for x

AS5 $A \Rightarrow (\forall x A)$ if x is not free in A

AS6 $(\forall v(A(v) \Rightarrow B(v)) \Rightarrow (\forall v A(v) \Rightarrow \forall v B(v))$

Inference rules:

- If A is a validity then $\forall v \ A$ is a validity
- Modus Ponens (inference rule from propositional logic)

Completeness Theorem

Theorem

A sentence of FOL (without equality) is valid iff it is provable in the previous system.

(Roughly) Gödel's Ph.D. thesis


Kurt Gödel 1906-1978

Corollary

The set of validities of FOL is CE

Again, turns out to be undecidable.

Prove satisfiability is undecidable (hence validity is...).
Encode non-halting (i.e. infinite, but well-formed) runs: variation of the proof for finite satisfiability.

Bad News on FO-Sat

Theorem

The problem of FO satisfiability is undecidable

Show NONHALT $\leq_m$ FOSAT

General idea: Trakhtenbrot's theorem took a TM + input and constructed FO sentence saying "this TM halts eventually".
Instead construct a sentence "this TM doesn't halt".
Negate clause that says "for some t, the t-th step contains TM in a halting state"...

Recall: Coding Big Runs by Sentences

Describe a run with a predicate logic world

Given any deterministic TM M and string w compute an FO sentence that describes a (code of) accepting run of M on w

Have predicates $\text{Time}(x)$, $\text{Space}(y)$, $\text{LessThan}(x_1, x_2)$
Have predicates $\text{HasSymbol}_a(t, s)$ and $\text{HasHead}_q(t, s)$ for every symbol a, state q

$\text{Time}(x)$ holds iff x is a number $\leq$ number of steps of machine
$\text{Space}(y)$ holds iff y is a number $\leq$ amount of space used by machine
$\text{LessThan}(x_1, x_2)$ holds iff x_1 and x_2 are both in Time or both in Space and x_1 comes before x_2
$\text{HasSymbol}_a(t, s)$ holds iff run at time t position s has symbol a
$\text{HasHead}_q(t, s)$ holds iff the head is on place s at step t of the run and state is q

Tape space j

		1	2	$\cdots$	s_{max}
Time	1	(q_0, w_1)	w_2	$\cdots$	
	2	w_1'	(q_1, w_2)		
	$\vdots$				
	$\vdots$				
	t_{final}				

This run corresponds to a world where
$\text{Time} = \{1, \ldots, t_{final}\}$
$\text{Space} = \{1, \ldots, s_{max}\}$
LessThan$=$ usual $<$ on numbers
$\text{HasSymbol}_{w_1} = \{(1, 1), \ldots\}$
$\text{HasHead}_{q_0} = \{(1, 1), \ldots\}$
$\text{HasSymbol}_{w_2} = \{(1, 2), \ldots\}$

Coding function describes a run of *arbitrary* length with a predicate logic world

Goal: Given any deterministic TM M and string w compute an FO sentence that describes a (code of) a non-halting run of M on w

Have predicates Time(x), Space(y), LessThan(x_1, x_2)
Have predicates HasSymbol$_a(t, s)$ and HasHead$_q(t, s)$ for every symbol a, state q

Time(x) holds iff x is a number $\leq$ number of steps of machine
Space(y) holds iff y is a number $\leq$ amount of space used by machine
LessThan(x_1, x_2) holds iff x_1 and x_2 are both in Time or both in Space and x_1 comes before x_2
HasSymbol$_a(t, s)$ holds iff run at time t position s has symbol a
HasHead$_q(t, s)$ holds iff the head is on place s at step t of the run and state is q

Tape space j

Time		1	2	...	...
	1	(q_0, w_1)	w_2	...	
	2	w_1'	(q_1, w_2)		
	$\vdots$				
	$\vdots$				
	$\vdots$				

Predicate Logic coding: This run corresponds to a world where
Time $= \{1, \ldots\}$
Space $= \{1, \ldots\}$
LessThan$=$ usual $<$ on numbers
HasSymbol$_{w_1}$ $= \{(1, 1), \ldots\}$
HasHead$_{q_0}$ $= \{(1, 1), \ldots\}$
HasSymbol$_{w_2}$ $= \{(1, 2), \ldots\}$

Undecidability of FO

Have all the axioms from before saying that the run starts with the initial state and satisfies the transition axioms.
Main change needed:

- Change the final state clause: say that configuration never gets to accepting or rejecting state.
  $\forall x \forall y \neg \mathrm{HasHead}_q(x, y)$ q is an accepting or rejecting state

Yes goes to Yes	M doesn't halt on w, take the infinite run & the corresponding world $\rightarrow$ $f(M, w)$ is satisfiable
No goes to No	If $f(M, w)$ is satisfiable, take the initial time, and closed under "next time" $\rightarrow$ this must be a non-halting run

Dealing with undecidability

reminder: "negative" results

- Model checking on a finite model is polynomial in the size of the model, **PSPACE**-complete in the query.
- Finite satisfiability checking (looking for a finite model) is CE but undecidable.
- Validity ("true over all models") is also CE, undecidable

One response: Use incomplete methods.

Given ϕ that you think is valid, use proof systems to try to search for a proof
$\rightarrow$ **Automatic Theorem Provers**

Given ϕ that you think is satisfiable, search for a finite model.
$\rightarrow$ **Model Finders**

Dealing with undecidability

(Satisfiability and Validity are undecidable.)

Response IIa) Restrict sentences:
There are fragments of FO (restricted kinds of sentences) for which satisfiability and/or validity are decidable.

Response IIb) Restrict **models**: there are classes of models, where the satisfiability problem relative to that set of models is decidable.
E.g. For the vocabulary $<(x, y)$ can restrict $<$ to be a linear order or another special kind of binary relation.

Response IIc) Restrict to a **particular** infinite model. A model M is said to be decidable if there is an algorithm that decides whether a sentence ϕ holds in M.
E.g. first-order theory of real numbers, or rational numbers.

Undecidability of FO theory of integers

Theorem

$M_{arith} = (Integers, +, *, <)$ *is undecidable*

That is, the language

$$\{\langle \phi \rangle \;:\; \mathrm{M}_{arith} \models \phi\}$$

is not decidable

Basic idea of proof: reduction from Halting or Acceptance problem. Encode runs of a Turing Machine by an integer.
new idea — an integer can represent a finite set. E.g. $2^3 3^5 7^{10}$ encodes the set $\{3, 5, 10\}$

Showing Models are Decidable

A collection of sentences A_{M} is a set of axioms for a model M if each sentence of A_{M} is true in M and A_{M} is complete:

for every sentence ϕ in vocabulary of M, either $A_{\mathsf{M}} \cup \phi$ is inconsistent or $A_{\mathsf{M}} \cup \neg\phi$ is inconsistent

That is, the logical consequences of A_{M} are the same as the sentences that hold in M.

Theorem

if M has a set of axioms that is C.E, then M is decidable (that is, we can decide which sentences hold in M)

E.g. $\mathsf{M}_{ratorder} = (\mathsf{Rationals}, <)$ has a complete finite set of axioms:
$<$ is a linear order (transitive, antisymmetric)
$<$ is dense: $\forall x \forall y \exists z \; x < z < y$
$<$ has no highest or lowest element
Hence, by the theorem, $\mathsf{M}_{ratorder}$ is decidable.

Decidability and Complete Axiomatization

Theorem

if M has a set of axioms that is C.E, then M is decidable (that is, we can decide which sentences hold in M)

Theorem

$M = (Integers, +, *, <)$ *is not decidable*

Corollary

for any c.e. collection of sentences A about $+, *, <$, if each element of A is true in the integers, then A must be incomplete: There is a sentence ϕ such that $A \cup \phi$ and $A \cup \neg\phi$ are both consistent

A weak form of Godel's Incompleteness Theorem

Summary: Logic and Universal Problems

Propositional Logic:

- model checking problems are linear time.
- satisfiability problems are decidable but **NP**-complete: "canonical hard problem"
- Proof systems can help make logic problems tractable in practice, but are not known to give polynomial worst-case bounds

First-Order Logic:

- model checking problem is **PSPACE**-complete, but "tractable in size of the model"
- (finite) satisfiability problems are semi/co semi-decidable
- Proof systems/theorem provers give semi-decidability of validity, can be useful in practice
- Can get decidability for
  - restricted classes of models (words, trees, graphs)
  - particular models (e.g. $\mathrm{M} = (N, +, <), M = (R, +, *, <)$)

Existential second-order logic

First-order properties are defined by FO sentences, e.g. in vocab of (directed) graphs:
$\forall v \neg \exists x, y\ E(v,x) \wedge E(v,y) \wedge x \neq y$ "out-degree ≤ 1"
$\forall x, y[E(x,y) \Rightarrow E(y,x)]\ \wedge\ \forall x, y, z\ [E(x,y) \wedge E(y,z) \Rightarrow E(x,z)]$
Now, allow quantification over *predicates* of specified arity.
"Existential": to keep things simple, just existential quantification over predicates.
"$\exists P(\cdot,\cdot)\phi$": there exists predicate P (of arity 2) such that ϕ

Evenness: the following formula is satisfied by interpretations for which the size of the domain is even (can't be expressed in FOPL):

$$\begin{aligned} \exists B, S \quad & \forall x \exists y B(x,y) \wedge \forall x, y, z B(x,y) \wedge B(x,z) \Rightarrow y = z \\ & \wedge \forall x, y, z B(x,z) \wedge B(y,z) \Rightarrow x = y \\ & \wedge \forall x, y S(x) \wedge B(x,y) \Rightarrow \neg S(y) \\ & \wedge \forall x, y \neg S(x) \wedge B(x,y) \Rightarrow S(y) \end{aligned}$$

(BTW there is no sentence ϕ of FOPL such that $\mathrm{I} \models \phi$ iff $|\mathrm{I}|$ is even.)

3-Colourability: The following formula is true in a graph (V, E) if and only if it is 3-colourable.

$$\begin{aligned}&\exists R, B, G \forall x(R(x) \vee B(x) \vee G(x))\\ &\qquad \forall x(\neg(R(x) \wedge B(x)) \wedge \neg(B(x) \wedge G(x)) \wedge \neg(R(x) \wedge G(x)) \wedge\\ &\qquad \forall x, y(E(x, y) \Rightarrow \neg(R(x) \wedge R(y)) \wedge \neg(B(x) \wedge B(y)) \wedge \neg(G(x) \wedge G(y)))\end{aligned}$$

Fagin's Theorem

Theorem

A class $\mathcal{C}$ *of finite structures (or interpretations) is definable by a sentence of* existential second-order logic *if and only if* $\mathcal{C}$ *is decidable by a non-deterministic TM running in polynomial time.*

So, we have another characterisation of the class **NP**.

$\Rightarrow$ ("only if"):

Given formula $\exists P_1, \ldots, P_r\ \phi$, can construct NTM M that, given interpretation I, guesses predicates $P_1, \ldots, P_r$ and checks them.

Runtime is exponential in arities of the P_i and in the depth of quantification in ϕ, but poly in $|\mathrm{I}|$ as required.

Fagin's Theorem

Theorem

A class $\mathcal{C}$ of finite structures (or interpretations) is definable by a sentence of existential second-order logic *if and only if $\mathcal{C}$ is decidable by a non-deterministic TM running in polynomial time.*

$\Leftarrow$ ("if") (much detail omitted):

NTM M having runtime n^k, that recognises instances of $\mathcal{C}$.

define $2k$-ary predicate "$<$": $\mathbf{x} < \mathbf{y}$ for $\mathbf{x}$ and $\mathbf{y}$ k-tuples of the domain of I.

k-ary predicates $S_q(\mathbf{x})$: the state of M at time $\mathbf{x}$ is q

$2k$-ary predicates $T_\sigma(\mathbf{x}, \mathbf{y})$: at time $\mathbf{x}$, the symbol at position $\mathbf{y}$ of the tape is σ

$2k$-ary predicate $H(\mathbf{x}, \mathbf{y})$: at time $\mathbf{x}$, tape head is located at position $\mathbf{y}$

$\exists <, S_q, T_\sigma, H\ \phi$: Clauses in ϕ to encode a run of M;

Define linear order $<$ on domain as before, then:

$$
<(x_1,\ldots,x_k,y_1,\ldots,y_k) \Leftrightarrow \quad
\begin{array}{l}
<(x_1,y_1) \\
(x_1=y_1)\wedge<(x_2,y_2) \\
(x_1=y_1)\wedge(x_2=y_2)\wedge<(x_2,y_2) \\
\ldots \\
(x_1=y_1)\wedge\ldots\wedge(x_{k-1}=y_{k-1})\wedge<(x_k,y_k)
\end{array}
$$