2.1 Predicate Logic

Consider the statement "Every student is younger than some instructor"

- We can't encode "every" and "some" in propositional logic
- Only choice is encode entire statement in an atomic proposition p
- Improvement: allow "atomic propositions" to be parametric over variables
  - Example: S(X) denotes "x is a student"
- Can reword it to fit it into predicate logic:
  - Reword to: For every x, if x is a student, then there exists a y such that y is an instructor and x is younger than y
  - This maps to $\forall$x(Student(x) $\implies$ $\exists$y(Instructor(y) $\land$ Younger(x,y)))
  - x and y don't have a type (they can represent any possible object)
  - Predicates are the only way to put constraints on variables

Soundness and Completeness of Predicate Logic

Will develop predicate calculus (similar to natural deduction) enabling syntactic reasoning with $\vdash$

Will develop semantic entailment for predicate logic, enabling semantic reasoning with $\models$

Similar to propositional logic, we have soundness and completeness of predicate logic:

$\phi_1, \phi_2, \ldots, \phi_\mathrm{n} \vdash \psi$ if and only if $\phi_1, \phi_2, \ldots, \phi_\mathrm{n} \models \psi$

P: predicate symbols

F: function symbols

Each predicate/function symbol has an **arity* denoting how many "parameters" it can apply to.

Constant (C): is a subset of functions ($C \subseteq F$) where the arity is 0. (No parameters)

Special predicate symbol $= \in P$ represnting equality

2.2 Predicate Logic as Formal Language

Terms

Terms represent the basic objects we want to reason about:

> term (t) ::= x | c | f(t,…,t)

- Variables x
- Constants c
- Function symbols f

Formulas

Formulas: a way of forming statements about the basic objects

> ϕ ::= P(t_1,t_2,..,t_n) | $\neg\phi$ | $\phi \land \phi$ | $\phi \lor \phi$ | $\phi \implies \phi$ (formula)
>
> $\forall$x(ϕ) | $\exists$x(ϕ)

Example

Encode: "Every daughter of my father is my sister"

- Represent "me": constant m
- Represent relationship "x is daughter of y": predicate D(x,y)
- Represent relationship "x is a father of y": predicate F(x,y)
- Represent relationshio "x is a sister of y": predicate S(x,y)
- Predicate logic encoding:

$$\forall x \forall y(F(x,m) \land D(y,x) \implies S(y,m))$$

Parsing Formulas

Operators precedence is similar to propositional logic

$$(\neg, \forall, \exists) \text{ (highest)}, \land, \lor, \implies$$

Implication ($\implies$) is right-associative

Parse Trees

Formula: $\forall x((P(x) \implies Q(x)) \land S(x,y))$

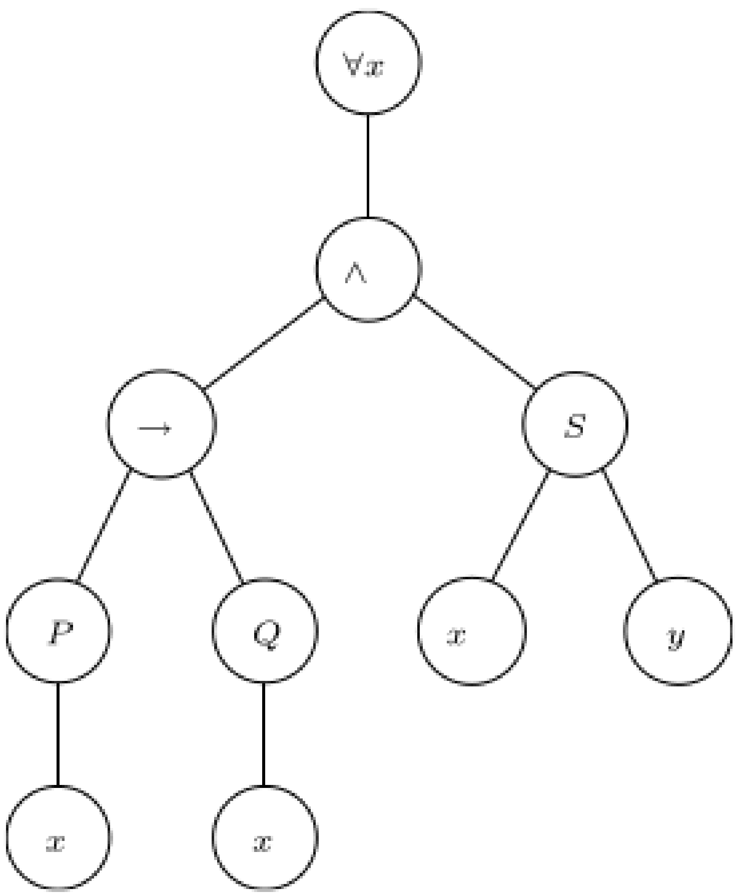


Figure 1: Parse tree with predicate

Free/Bound Variables and Substitution

Quantifiers ∀x, ∃x introduce a *scope* for x (any child after ∀x,with an x, will be referencing the original x in the ∀)

Quantified variables are bound, otherwise free.

Use notation ϕ[t/x] to denote the formula produced by replacing all free occurences of x with t in ϕ

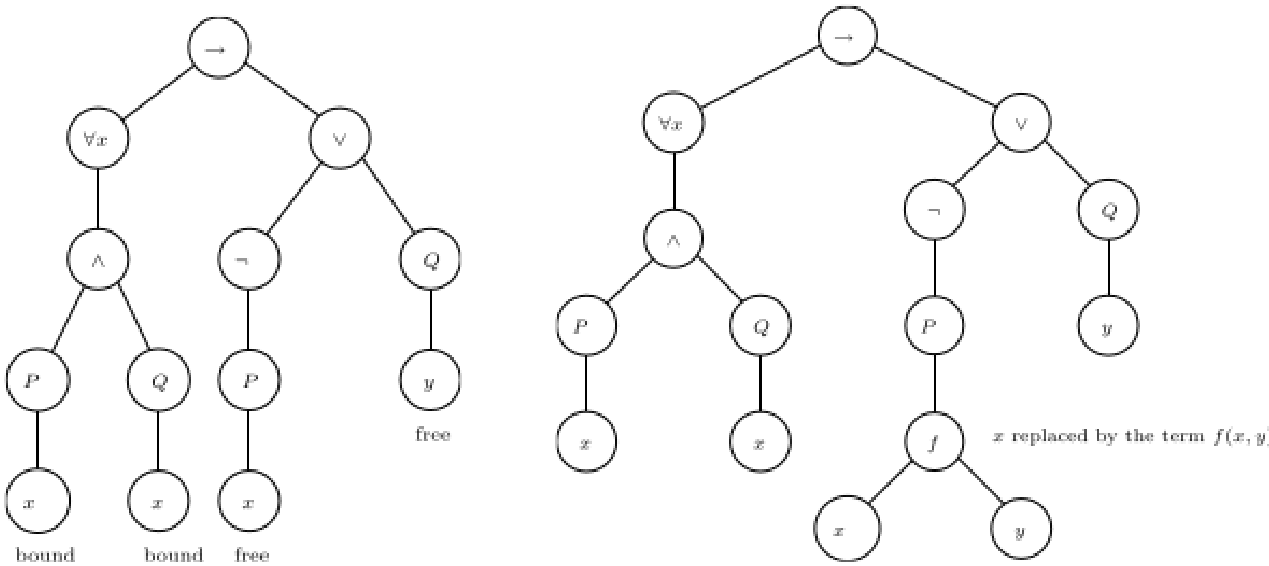
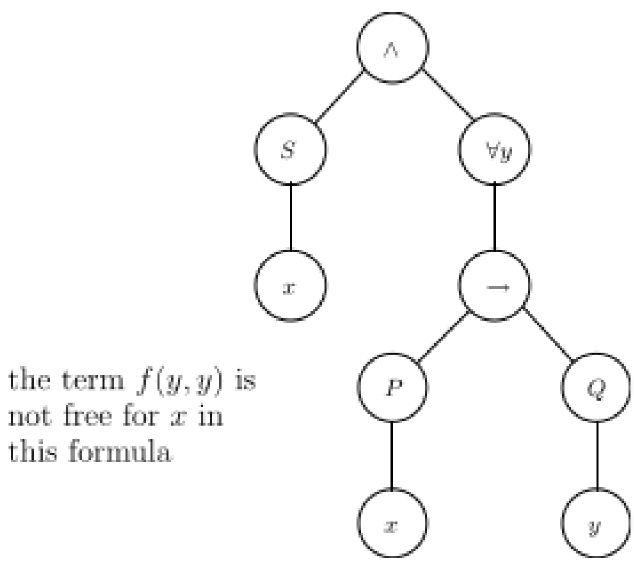


Figure 2: Tree on the left shows bound variables and scope, Tree on the right shows replacing of free variables

- ϕ[f(x,y)/x]: substitution performed on right tree.

Capture-avoiding substitution



If you were to perform ϕ[f(y,y)/x] on the parse tree above, that would break.

Capture avoidance - need to use a fresh variable y': ϕ[f(y',y')/x]

2.3 Proof Theory of Predicate Logic

Need calculus like natural deduction that let us derive conclusions from premises

Will see that

$\neg\forall x(\phi) \dashv\vdash \exists x(\neg\phi)$

Extending Propositional Logic

Propositional logic = predicate logic limited to 0-ary predicates

Therefore, all we need to full represent predicate is add some extension:

- Terms (specifically, equality between terms)
- Universal quantifiers $\forall$
- Existential quantifiers $\exists$

Rules for Equality

EQ-INTRO (Introducing =): $\dfrac{}{t = t}$

EQ-ELIM (Eliminating =): $\dfrac{t_1 = t_2 \quad \phi[t_1/x]}{\phi[t_2/x]}$

- Can choose however many terms you want to replace with elimination

Example: Equality elimination

$x+1 = 1+x, (x+1 > 1) \rightarrow (x+1 > 0) \vdash (1+x) > 1 \rightarrow (1+x) > 0$

Step	Formula	Rule
1	$(x+1) = (1+x)$	premise
2	$(x+1 > 1) \rightarrow (x+1 > 0)$	premise
3	$(1+x > 1) \rightarrow (1+x > 0)$	=e 1,2

Example: Equality introduction

$t_1 = t_2 \vdash t_2 = t_1$

Step	Formula	Rule
1	$t_1 = t_2$	premise
2	$t_1 = t_1$	=i
3	$t_2 = t_1$	=e 1,2

$t_1 = t_2, t_2 = t_3 \vdash t_1 = t_3$

Step	Formula	Rule
1	$t_2 = t_3$	premise
2	$t_1 = t_2$	premise
3	$t_1 = t_3$	=e 1,2

Rules for Universal Quantification

ALL-ELIM (Eliminating $\forall$): $\dfrac{\forall x(\phi)}{\phi[t/x]}$

ALL-INTRO (Introducing $\forall$): $\dfrac{<x_0> \ldots\ \phi[x_0/x]}{\forall x(\phi)}$

$<x_0>$ = fresh variables

Example: $\forall$ elimination

P(t), $\forall$x(P(x) $\rightarrow$ $\neg$Q(x)) $\vdash$ $\neg$Q(t)

Step	Formula	Rule
1	P(t)	premise
2	$\forall$x(P(x) $\rightarrow$ $\neg$Q(x))	premise
3	P(t) $\rightarrow$ $\neg$Q(t)	$\forall$x e 2
4	$\neg$Q(t)	$\rightarrow$e 3,1

Example $\forall$ introduction

$\forall$x(P(x) $\rightarrow$ Q(x)), $\forall$xP(x) $\vdash$ $\forall$xQ(x)

Step		Formula	Rule
1		$\forall$x(P(x) $\rightarrow$ Q(x))	premise
2		$\forall$xP(x)	premise
3	x_0	P(x_0) $\rightarrow$ Q(x_0)	$\forall$x e 1
4		P(x_0)	$\forall$x e 2
5		Q(x_0)	$\rightarrow$e 3,4
6		$\forall$xQ(x)	$\forall$x i 3-5

Rule for Existential Quantification

EX-INTRO (Introduce $\exists$): $\dfrac{\phi[t/x]}{\exists x(\phi)}$

EX-ELIM (Eliminating $\exists$): $\dfrac{\exists x(\phi) \qquad <x_0>\ \phi[x_0/x]\ldots\chi}{\chi}$

Example: $\exists$ introduction

$\forall$xϕ $\vdash$ $\exists$xϕ

Step	Formula	Rule
1	$\forall$xϕ	premise
2	ϕ[x/x]	$\forall$x e 1
3	$\exists$xϕ	$\exists$x i 2

Example: $\exists$ elimination

$\forall x(P(x) \rightarrow Q(x)), \exists x P(x) \vdash \exists x Q(x)$

Step		Formula	Rule
1		$\forall x(P(x) \rightarrow Q(x))$	premise
2		$\exists x P(x)$	premise
3	x_0	$P(x_0)$	assumption
4		$P(x_0) \rightarrow Q(x_0)$	$\forall$x e 1
5		$Q(x_0)$	$\rightarrow$e 4,3
6		$\exists x Q(x)$	$\exists$x i 5
7		$\exists x Q(x)$	$\exists$x e 2,3-6

Quantifier Equivalences

$\neg\forall x(\phi) \dashv\vdash \exists x(\neg\phi)$

$\neg\exists x(\phi) \dashv\vdash \forall x(\neg\phi)$

Other equivalences:

- $\forall x\phi \wedge \forall x\psi \dashv\vdash \forall x(\phi \wedge \psi)$
- $\exists x\phi \vee \exists x\psi \dashv\vdash \exists x(\phi \vee \psi)$

- $\forall x\forall y\phi \dashv\vdash \forall y\forall x\phi$
- $\exists x\exists y\phi \dashv\vdash \exists y\exists x\phi$

If x is not free in ψ:

- $\forall x\ \phi \wedge \psi \dashv\vdash \forall x(\phi \wedge \psi)$
- $\forall x\ \phi \vee \psi \dashv\vdash \forall x(\phi \vee \psi)$
- $\exists x\ \phi \wedge \psi \dashv\vdash \exists x(\phi \wedge \psi)$
- $\exists x\ \phi \vee \psi \dashv\vdash \exists x(\phi \vee \psi)$
- $\forall x(\psi \rightarrow \phi) \dashv\vdash \psi \rightarrow \forall x\psi$
- $\exists x(\psi \rightarrow \phi) \dashv\vdash \psi \rightarrow \exists x\psi$
- $\exists x(\phi \rightarrow \psi) \dashv\vdash \forall x\phi \rightarrow \psi$
- $\forall x(\phi \rightarrow \psi) \dashv\vdash \exists x\phi \rightarrow \psi$

Clever Trick for Proofs

Think of $\forall$ as $\wedge$

Think of $\exists$ as $\vee$

To prove $\neg\forall x(\phi) \dashv\vdash \exists x(\neg\phi)$

- First prove $\neg(p_1 \wedge p_2) \dashv\vdash (\neg p_1 \vee \neg p_2)$

2.4 Semantics of Predicate Logic

Models

Need to extend the propositional logic concept of "valuation"

- Terms (objects that can be many things) need to be mapped to concrete objects
- Function symbols need to be mapped to concrete functions from objects to objects
- Predicate symbols need to be mapped to concrete relations over objects

Model M of (F,P). F = function symbol, P = predicate symbol:

- Nonempty set A of concrete objects
- Concrete object $\mathrm{g}^M \in \mathrm{A}$ for each 0-arity function symbol g (constant)
- Concrete function f^M: $\mathrm{A}^\mathrm{n} \to \mathrm{A}$ for each n-arity function symbol f
- Concrete relation $\mathrm{P}^M \subseteq \mathrm{A}^\mathrm{n}$ for each predicate symbol P

Symbol^M means symbol is part of the model M

Example:

- Concrete objects (states): $\mathrm{A} = \{\mathrm{a,b,c}\}$
- Constant (initial state): $\mathrm{i\hat{}}M = \mathrm{a}$
- Predicate symbol (transition relation): $\mathrm{R}^M = \{(\mathrm{a,a}),(\mathrm{a,b}),(\mathrm{a,c}),(\mathrm{b,c}),(\mathrm{c,c})\}$
  - There's some state that is reachable from the initial state
- Predicate symbol (final states): $\mathrm{F}^M = \{\mathrm{b,c}\}$
- Now we can give meaning to formulas:
  - $\exists$y(R(i,y)): y is reachable from the initial state
  - $\neg$F(i): the initial state is not a final state
  - $\forall$x$\exists$y(R(x,y)): For every state, there's a different state that reach any state.

Example (textbook):

- Given a model M for a pair (F,P) and given an environment l, we define the satisfaction relation $M \models l\phi$ for each logical formula ϕ over the pair (F,P) and look-up table l by structural induction on ϕ. If $M \models l\phi$ holds, we say that ϕ computes to T in the model M with respect to the environment l.
- P: If ϕ is of the form $\mathrm{P}(\mathrm{t}_1,\mathrm{t}_2,\ldots,\mathrm{t}_\mathrm{n})$, then we interpret the terms $\mathrm{t}_1,\mathrm{t}_2,\ldots,\mathrm{t}_\mathrm{n}$ in our set A by replacing all the variables with their values according to l. In this way we compute concrete values $\mathrm{a}_1,\mathrm{a}_2,\ldots,\mathrm{a}_\mathrm{n}$ of A for each of these terms, where we interpret any function symbol $\mathrm{f} \in F$ by f^M. Now $M \models l\mathrm{P}(\mathrm{t}_1,\mathrm{t}_2,\ldots,\mathrm{t}_\mathrm{n})$ holds iff $(\mathrm{a}_1,\mathrm{a}_2,\ldots,\mathrm{a}_\mathrm{n})$ is in set P^M
  - l = mapping, universe, every bullet from previous example
  - M $\models l\phi$ is for a phi the universe/mapping makes the model true
  - P: apply l environment on concrete objects and thus plug that in into relation P and see if it holds
  - $\forall$ and $\exists$: see if there's something in the mapping where it holds
  - $\neg$, $\wedge$, $\vee$, $\implies$: see if they hold in the mapping
- This is the definition of model in the textbook page 128. It is formal AF, don't think you'll need to know this but idk.

Semantic Entailment and Satisfiability

Given set of formulas Γ:

$\Gamma \models \psi$ if and only if for all models M and lookup tables l, whenever $M \models l\phi$ for all $\phi \in \Gamma$, then $M \models l\psi$

ψ is satisfiable if and only if there exists some model M and lookup table l such that $M \models l\psi$

Theories

Theories give a specific meaning to predicate/function symbols

Theory of equality and uninterpreted function symbols (EUF):

$F = \{\text{f}, \ldots\}$ (arbitrary function), $P = \{=\}$ (equal predicate)

Theory of Presburger arithmetic:

$F = \{0,1,+,-\}$, $P = \{\leq\}$

Satisfiability Modulo Theories (SMT)

Example (Checking satisfiability):

- Function: $\{a * (f(b) + f(c)) = d, b(f(a) + f(c)) \neq d, a = b\}$
  - Convert times to h
  - Convert plus to g
- New function: $\{h(a,g(f(b),f(c))) = d, h(b,g(f(a),f(c))) \neq d, a = b\}$
  - Already unsatisfiable just using EUF

Lazy SMT

Lazy SMT = DPLL + theory solver

Check satisfiability (Lazy SMT): $\{a < b \implies a < c, b < c, a < b\}$

- Encode theory expressions using propositions
  - $p \equiv a < b$
  - $q \equiv a < c$
  - $r \equiv b < c$
- Run DPLL: $\{p \implies q,r,p\} = \{\neg p \vee q, r, p\}$
- Discover satisfying assignment: $p = \top$, $q = \top$, $r = \top$
- Check satisfiability using *theory solver*: $\{p,q,r\} = \{a<b, a<c, b<c\}$
- Theory solver finds satisfying assignment: a=1, b=2, c=3