4.

Resolution

Goal

Deductive reasoning in language as close as possible to full FOL

$\neg, \ \wedge, \ \vee, \ \exists, \forall$

Knowledge Level:

given KB, α, determine if KB |= α.

or given an open $\alpha[x_1,x_2,...x_n]$, find $t_1,t_2,...t_n$ such that KB |= $\alpha[t_1,t_2,...t_n]$

When KB is finite $\{\alpha_1, \alpha_2, ..., \alpha_k\}$

KB |= α

iff |= $[(\alpha_1 \wedge \alpha_2 \wedge ... \wedge \alpha_k) \supset \alpha]$

iff KB $\cup \{\neg\alpha\}$ is unsatisfiable

iff KB $\cup \{\neg\alpha\}$ |= FALSE

where FALSE is something like $\exists x.(x \neq x)$

So want a procedure to test for validity, or satisfiability, or for entailing FALSE.

Will now consider such a procedure (first without quantifiers)

Clausal representation

Formula = set of clauses

Clause = set of literals

Literal = atomic sentence or its negation

positive literal and negative literal

Notation:

If ρ is a literal, then $\bar{\rho}$ is its complement

$\overline{p} \Rightarrow \neg p \qquad \overline{\neg p} \Rightarrow p$

To distinguish clauses from formulas:

[and] for clauses: $[p, \overline{r}, s]$ { and } for formulas: $\{ [p, \overline{r}, s], [p, r, s], [\overline{p}] \}$

[] is the empty clause {} is the empty formula

So {} is different from {[]}!

Interpretation:

Formula understood as <u>conjunction</u> of clauses

Clause understood as <u>disjunction</u> of literals

Literals understood normally

$\{[p, \neg q], [r], [s]\}$ represents $((p \vee \neg q) \wedge r \wedge s)$

[ ] represents FALSE

CNF and DNF

Every propositional wff α can be converted into a formula α' in Conjunctive Normal Form (CNF) in such a way that $\models \alpha \equiv \alpha'$.

1. eliminate $\supset$ and $\equiv$ using $(\alpha \supset \beta) \twoheadrightarrow (\neg\alpha \vee \beta)$ etc.
2. push $\neg$ inward using $\neg(\alpha \wedge \beta) \twoheadrightarrow (\neg\alpha \vee \neg\beta)$ etc.
3. distribute $\vee$ over $\wedge$ using $((\alpha \wedge \beta) \vee \gamma) \twoheadrightarrow ((\alpha \vee \gamma) \wedge (\beta \vee \gamma))$
4. collect terms using $(\alpha \vee \alpha) \twoheadrightarrow \alpha$ etc.

Result is a conjunction of disjunction of literals.

an analogous procedure produces DNF,
a disjunction of conjunction of literals

We can identify CNF wffs with clausal formulas

$$(p \vee \neg q \vee r) \wedge (s \vee \neg r) \twoheadrightarrow \{ [p, \neg q, r], [s, \neg r] \}$$

So: given a finite KB, to find out if KB $\models \alpha$, it will be sufficient to

1. put $(KB \wedge \neg\alpha)$ into CNF, as above
2. determine the satisfiability of the clauses

Resolution rule of inference

Given two clauses, infer a new clause:

From clause $\{ p \} \cup C_1$,

and $\{\neg p \} \cup C_2$,

infer clause $C_1 \cup C_2$.

$C_1 \cup C_2$ is called a <u>resolvent</u> of input clauses with respect to p.

Example:

clauses $[w, r, q]$ and $[w, s, \neg r]$ have $[w, q, s]$ as resolvent wrt r.

Special Case:

$[p]$ and $[\neg p]$ resolve to $[]$ (the C_1 and C_2 are empty)

A <u>derivation</u> of a clause c from a set S of clauses is a sequence $c_1, c_2, ..., c_n$ of clauses, where $c_n = c$, and for each c_i, either

1. $c_i \in S$, or
2. c_i is a resolvent of two earlier clauses in the derivation

Write: $S \rightarrow c$ if there is a derivation

Rationale

Resolution is a symbol-level rule of inference, but has a connection to knowledge-level logical interpretations

Claim: Resolvent is <u>entailed</u> by input clauses.

Suppose $\Im \models (p \vee \alpha)$ and $\Im \models (\neg p \vee \beta)$

Case 1: $\Im \models p$

then $\Im \models \beta$, so $\Im \models (\alpha \vee \beta)$.

Case 2: $\Im \not\models p$

then $\Im \models \alpha$, so $\Im \models (\alpha \vee \beta)$.

Either way, $\Im \models (\alpha \vee \beta)$.

So: $\{(p \vee \alpha), (\neg p \vee \beta)\} \models (\alpha \vee \beta)$.

Special case:

$[p]$ and $[\neg p]$ resolve to [ ],

so $\{[p], [\neg p]\} \models$ FALSE

that is: $\{[p], [\neg p]\}$ is unsatisfiable

Derivations and entailment

Can extend the previous argument to derivations:

If $S \rightarrow c$ then $S \models c$

Proof: by induction on the length of the derivation.
Show (by looking at the two cases) that $S \models c_i$.

But the converse does not hold in general

Can have $S \models c$ without having $S \rightarrow c$.

Example: $\{[\neg p]\} \models [\neg p, \neg q]$ i.e. $\neg p \models (\neg p \vee \neg q)$
but no derivation

However.... Resolution *is* <u>refutation complete</u>!

Theorem: $S \rightarrow []$ iff $S \models []$

sound and complete
when restricted to [ ]

Result will carry over to quantified clauses (later)

So for any set S of clauses: S is unsatisfiable iff $S \rightarrow []$.

Provides method for determining satisfiability: search all derivations for [].

So provides a method for determining all entailments

A procedure for entailment

To determine if KB |= α,

- put KB, $\neg\alpha$ into CNF to get *S*, as before
- check if $S \rightarrow$ [].

If KB = {}, then we are testing the validity of α

Non-deterministic procedure

1. Check if [] is in *S*.

   If yes, then return **UNSATISFIABLE**

2. Check if there are two clauses in *S* such that they resolve to produce a clause that is not already in *S.*

   If no, then return **SATISFIABLE**

3. Add the new clause to *S* and go to 1.

Note: need only convert KB to CNF once

- can handle multiple queries with same KB
- after addition of new fact α, can simply add new clauses α' to KB

So: good idea to keep KB in CNF

Example 1

KB

FirstGrade

FirstGrade $\supset$ Child

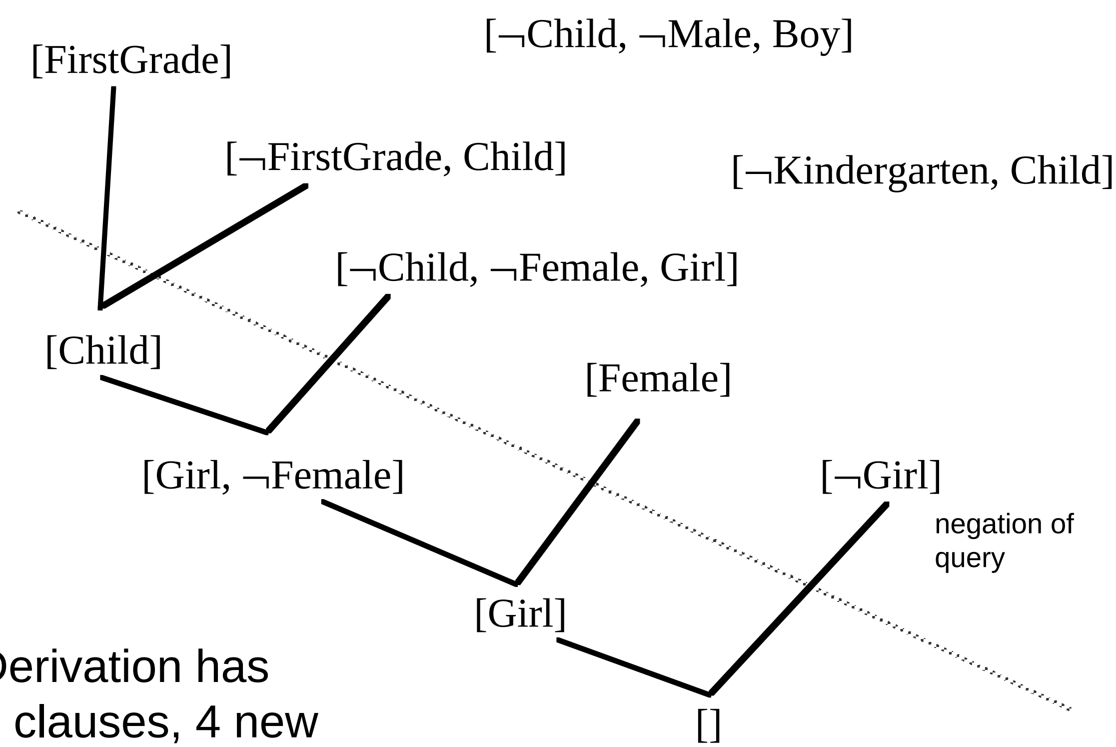
Child $\wedge$ Male $\supset$ Boy

Kindergarten $\supset$ Child

Child $\wedge$ Female $\supset$ Girl

Female

Show that KB |= Girl



Derivation has
9 clauses, 4 new

Example 2

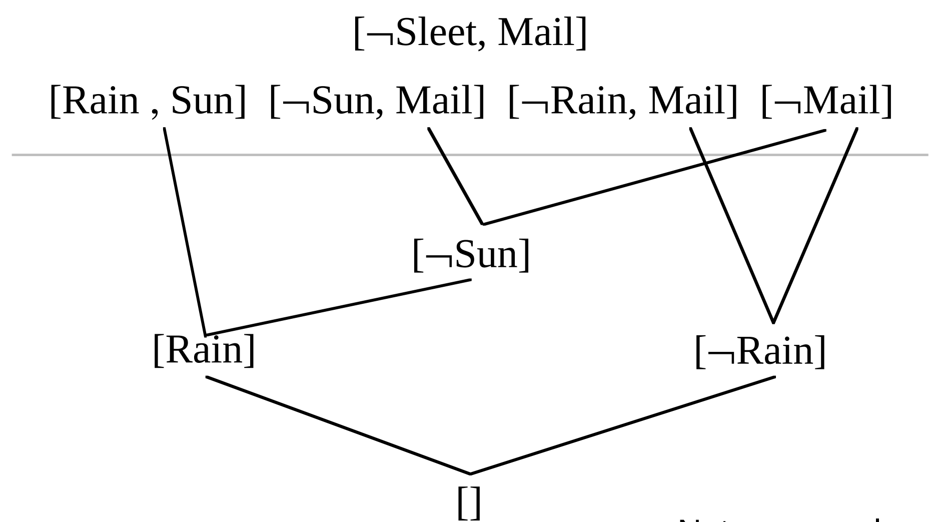
KB

(Rain ∨ Sun)

(Sun ⊃ Mail)

((Rain ∨ Sleet) ⊃ Mail)

Show KB |= Mail



Note: every clause not in *S* has 2 parents

Similarly KB |≠ Rain

Can enumerate all resolvents given ¬Rain, and [] will not be generated

Quantifiers

Clausal form as before, but atom is $P(t_1, t_2, ..., t_n)$, where t_i may contain variables

Interpretation as before, but variables are understood *universally*

Example: $\{ [P(x), \neg R(a,f(b,x))], [Q(x,y)] \}$

interpreted as

$\forall x \forall y \{[R(a,f(b,x)) \supset P(x)] \wedge Q(x,y)\}$

Substitutions: $\theta = \{v_1/t_1, v_2/t_2, ..., v_n/t_n\}$

Notation: If ρ is a literal and θ is a substitution, then $\rho\theta$ is the result of the substitution (and similarly, $c\theta$ where c is a clause)

Example: $\theta = \{x/a, y/g(x,b,z)\}$

$P(x,z,f(x,y))\,\theta = P(a,z,f(a,g(x,b,z)))$

A literal is <u>ground</u> if it contains no variables.

A literal ρ is an <u>instance</u> of ρ', if for some θ, $\rho = \rho'\theta$.

Generalizing CNF

Resolution will generalize to handling variables

Ignore = for now

But to convert wffs to CNF, we need three additional steps:

1. eliminate $\supset$ and $\equiv$

2. push $\neg$ inward using also $\neg\forall x.\alpha \Rightarrow \exists x.\neg\alpha$ etc.

3. standardize variables: each quantifier gets its own variable

   e.g. $\exists x[P(x)] \wedge Q(x) \Rightarrow \exists z[P(z)] \wedge Q(x)$ where z is a new variable

4. eliminate all existentials *(discussed later)*

5. move universals to the front using $(\forall x\alpha) \wedge \beta \Rightarrow \forall x(\alpha \wedge \beta)$

   where β does not use x

6. distribute $\vee$ over $\wedge$

7. collect terms

Get universally quantified conjunction of disjunction of literals

then drop all the quantifiers...

First-order resolution

Main idea: a literal (with variables) stands for all its instances; so allow all such inferences

So given $[P(x,a), \neg Q(x)]$ and $[\neg P(b,y), \neg R(b,f(y))]$,
want to infer $[\neg Q(b), \neg R(b,f(a))]$ among others

since $[P(x,a), \neg Q(x)]$ has $[P(b,a), \neg Q(b)]$ and
$[\neg P(b,y), \neg R(b,f(y))]$ has $[\neg P(b,a), \neg R(b,f(a))]$

Resolution:

Given clauses: $\{\rho_1\} \cup C_1$ and $\{\overline{\rho_2}\} \cup C_2$.

Rename variables, so that distinct in two clauses.

For any θ such that $\rho_1\theta = \rho_2\theta$, can infer $(C_1 \cup C_2)\theta$.

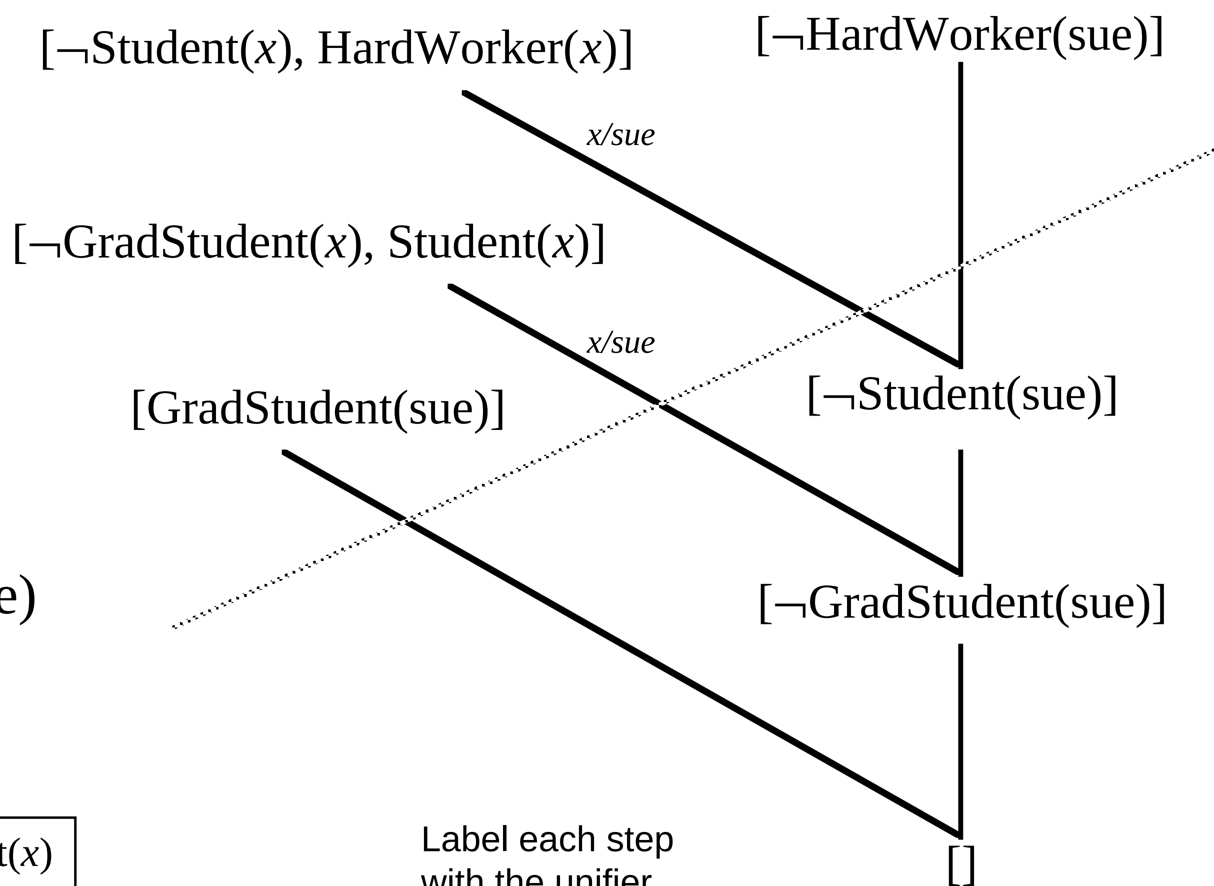
We say that ρ_1 <u>unifies</u> with ρ_2 and that θ is a <u>unifier</u> of the two literals

Resolution derivation: as before

Theorem: $S \rightarrow []$ iff $S \models []$ iff S is unsatisfiable

Note: There are pathological examples where a slightly more general definition of Resolution is required. We ignore them for now...

Example 3



?
KB |= HardWorker(sue)

KB

$\forall x$ GradStudent(x) $\supset$ Student(x)
$\forall x$ Student(x) $\supset$ HardWorker(x)
GradStudent(sue)

Label each step with the unifier

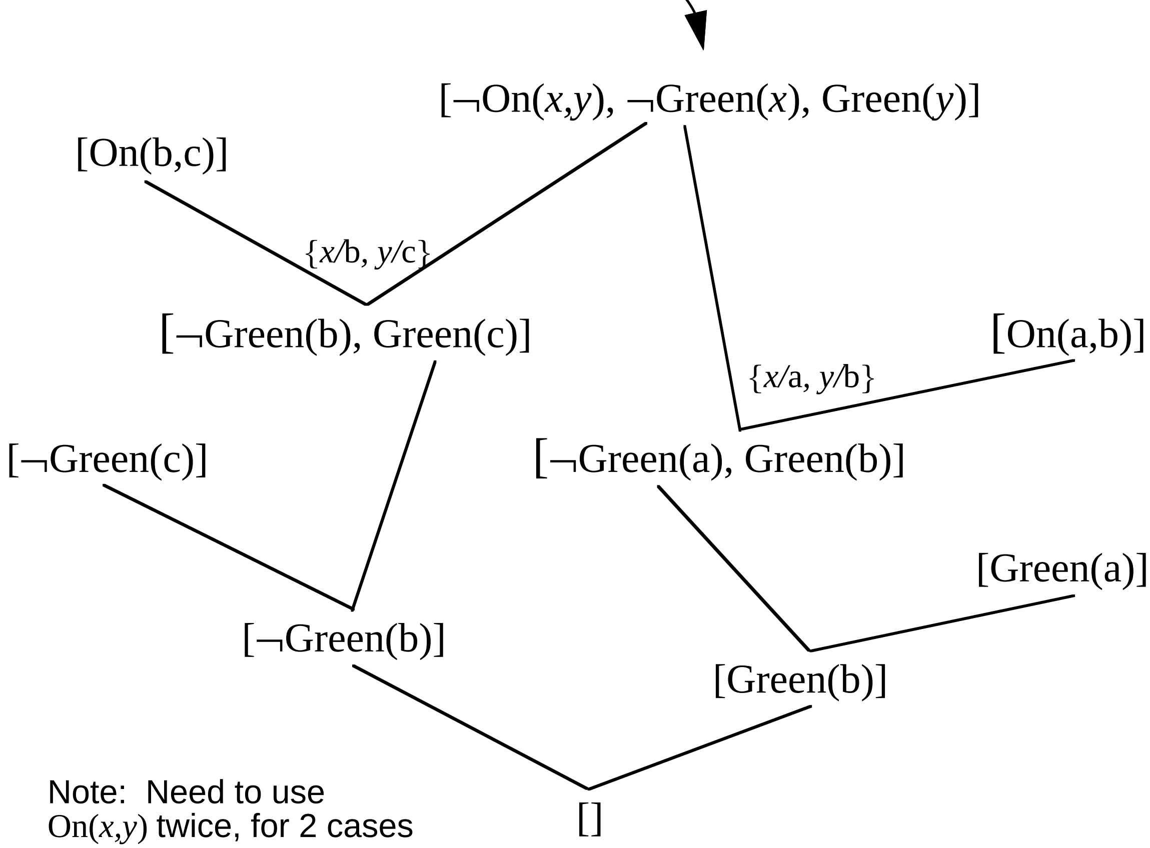
Point to relevant literals in clauses

The 3 block example

KB = {On(a,b), On(b,c), Green(a), ¬Green(c)}   already in CNF

Query = $\exists x \exists y[\mathrm{On}(x,y) \wedge \mathrm{Green}(x) \wedge \neg\mathrm{Green}(y)]$

Note: ¬Q has no existentials, so yields



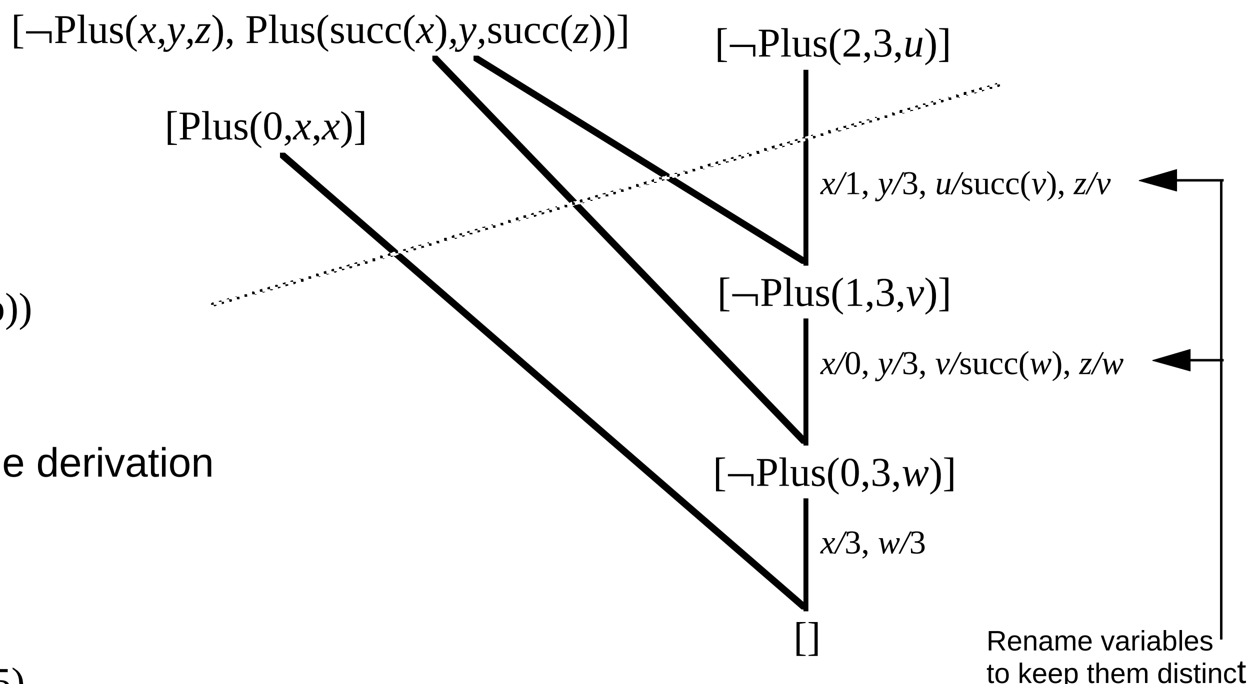
Note: Need to use On(x,y) twice, for 2 cases

Arithmetic

KB: Plus(zero,x,x)
Plus(x,y,z) $\supset$ Plus(succ(x),y,succ(z))

Q: $\exists u$ Plus(2,3,u)

For readability,
we use
0 for zero,
1 for succ(zero),
2 for succ(succ(zero))
etc.



Can find the answer in the derivation

u/succ(succ(3))

that is: u/5

Can also derive Plus(2,3,5)

Skolemization

So far, converting wff to CNF ignored existentials

e.g. $\exists x \forall y \exists z P(x,y,z)$

Idea: names for individuals claimed to exist, called <u>Skolem</u> constant and function symbols

there exists an x, call it a

for each y, there is a z, call it $f(y)$

get $\forall y P(a,y,f(y))$

So replace $\forall x_1(\ldots\forall x_2(\ldots\forall x_n(\ldots\exists y[\ldots\ y\ \ldots]\ \ldots)\ldots)\ldots)$

by $\forall x_1(\ldots\forall x_2(\ldots\forall x_n(\ \ldots\ \ [\ldots\ f(x_1,x_2,\ldots,x_n)\ \ldots]\ \ldots)\ldots)\ldots)$

f is a new function symbol that appears nowhere else

Skolemization does <u>not</u> preserve equivalence

e.g. $\not\models\ \exists x P(x) \equiv P(a)$

But it does preserve satisfiability

α is satisfiable iff α' is satisfiable (where α' is the result of Skolemization)

sufficient for resolution!

Variable dependence

Show that $\exists x \forall y R(x,y) \models \forall y \exists x R(x,y)$

show $\{\exists x \forall y R(x,y),\ \neg\forall y \exists x R(x,y)\}$ unsatisfiable

$\exists x \forall y R(x,y) \twoheadrightarrow \forall y R(a,y)$

$\neg\forall y \exists x R(x,y) \twoheadrightarrow \exists y \forall x \neg R(x,y) \twoheadrightarrow \forall x \neg R(x,b)$

then $\{\ [R(a,y)],\ [\neg R(x,b)]\ \} \rightarrow []$ with $\{x/a,\ y/b\}$.

Show that $\forall y \exists x R(x,y) \not\models \exists x \forall y R(x,y)$

show $\{\forall y \exists x R(x,y),\ \neg\exists x \forall y R(x,y)\}$ satisfiable

$\forall y \exists x R(x,y) \twoheadrightarrow \forall y R(f(y),y)$

$\neg\exists x \forall y R(x,y) \twoheadrightarrow \forall x \exists y \neg R(x,y) \twoheadrightarrow \forall x \neg R(x,g(x))$

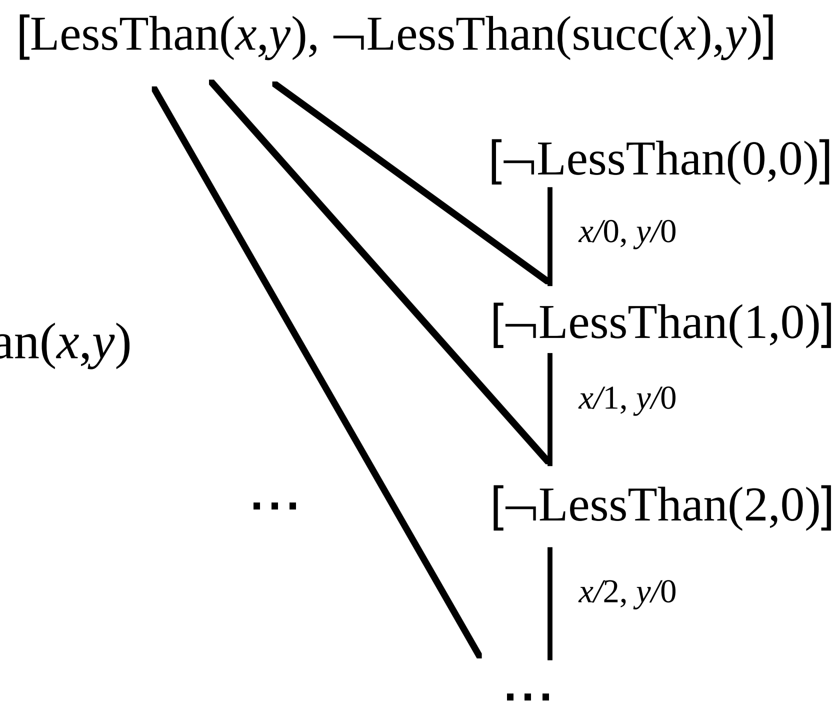
then get $\{\ [R(f(y),y)],\ [\neg R(x,g(x))]\ \}$

where the two literals do <u>not</u> unify

Note: important to get dependence of variables correct

$R(f(y),y)$ vs. $R(a,y)$ in the above

A problem



KB:

LessThan(succ(x),y) $\supset$ LessThan(x,y)

Query:

LessThan(zero,zero)

Should fail since KB $|\neq$ Q

Infinite branch of resolvents

cannot use a simple depth-first procedure to search for []

Undecidability

Is there a way to detect when this happens?

No! FOL is very powerful

can be used as a full programming language

just as there is no way to detect in general when
a program is looping

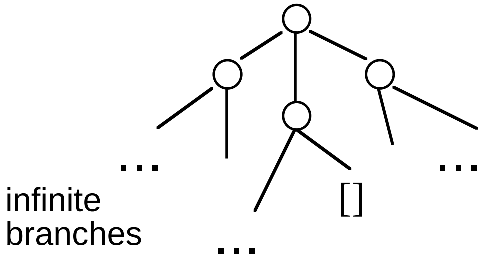
There can be no procedure that does this:

Proc[*Clauses*] =

If *Clauses* are unsatisfiable
then return YES
else return NO

However: Resolution is complete

some branch will contain [], for unsatisfiable clauses



So breadth-first search guaranteed to find []

search may not terminate on satisfiable clauses

Overly specific unifiers

In general, no way to guarantee efficiency, or even termination

later: put control into users' hands

One thing that can be done:

reduce redundancy in search, by keeping search as general as possible

Example

$$\ldots, P(g(x),f(x),z)] \quad [\neg P(y,f(w),a), \ldots$$

unified by

$\theta_1 = \{x/b,\ y/g(b),\ z/a,\ w/b\}$ gives $P(g(b),f(b),a)$

and by

$\theta_2 = \{x/f(z),\ y/g(f(z)),\ z/a,\ w/f(z)\}$ gives $P(g(f(z)),f(f(z)),a)$.

Might not be able to derive the empty clause from clauses having overly specific substitutions

wastes time in search!

Most general unifiers

θ is a <u>most general unifier</u> (MGU) of literals ρ_1 and ρ_2 iff

1. θ unifies ρ_1 and ρ_2
2. for any other unifier θ', there is a another substitution θ^* such that $\theta' = \theta\theta^*$

Note: composition $\theta\theta^*$ requires applying θ^* to terms in θ

for previous example, an MGU is

$$\theta = \{x/w,\ y/g(w),\ z/a\}$$

for which

$$\theta_1 = \theta\{w/b\}$$

$$\theta_2 = \theta\{w/f(z)\}$$

Theorem: Can limit search to most general unifiers only without loss of completeness (with certain caveats)

Computing MGUs

Computing an MGU, given a set of literals $\{\rho_i\}$

usually only have two literals

1. Start with $\theta := \{\}$.
2. If all the $\rho_i\theta$ are identical, then done;
   otherwise, get disagreement set, *DS*

   e.g $P(a,f(a,g(z),\ldots \quad P(a,f(a,u,\ldots$

   disagreement set, $DS = \{u, g(z)\}$
3. Find a variable $v \in DS$, and a term $t \in DS$ not containing v.
   If not, fail.
4. $\theta := \theta\{v/\,t\}$
5. Go to 2

Note: there is a better *linear* algorithm

Resolution is difficult!

First-order resolution is not guaranteed to terminate.

What can be said about the propositional case?

- Shown by Haken in 1985 that there are unsatisfiable clauses $\{c_1, c_2, ..., c_n\}$ such that the *shortest* derivation of [] contains on the order of 2^n clauses
  - Even if we could always find a derivation immediately, the most clever search procedure will still require *exponential* time on some problems

Problem just with resolution?

- Probably not.
- Determining if a set of clauses is satisfiable was shown by Cook in 1972 to be NP-complete
  - No easier than an extremely large variety of computational tasks
  - Roughly: any search task where what is searched for can be verified in polynomial time can be recast as a satisfiability problem
    - » satisfiability
    - » does graph of cities allow for a full tour of size $\leq k$ miles?
    - » can N queens be put on an N$\times$N chessboard all safely? and many, many more....
- Satisfiability is believed by most people to be unsolvable in polynomial time