Resolution – answer extraction

It is often possible to get answers to questions by looking at the bindings of variables in a derivation of an existential (*Plus* predicate in Example 5 in the previous course).

But more complicated situations may appear in FOL. In the three-block problem (Example 4 in the previous course), we know that there is a block that satisfies a condition, but we don't know which block.

KB: On(a,b), On(b,c), Green(a), ¬Green(c)

Question: ∃x∃y.Green(x)∧¬Green(y)∧On(x,y)

That is to say that KB ⊨ ∃x.P(x) without entailing P(t) for a specific t.

These slides are mainly based on Ronald J. Brachman, Hector J. Levesque. Knowledge representation and reasoning, Morgan Kaufmann, 2004.

Resolution – answer extraction

Idea: replace a question such as ∃x.P(x) by ∃x.P(x)∧¬A(x), where A is a new predicate symbol that occurs nowhere else. A is called the answer predicate.

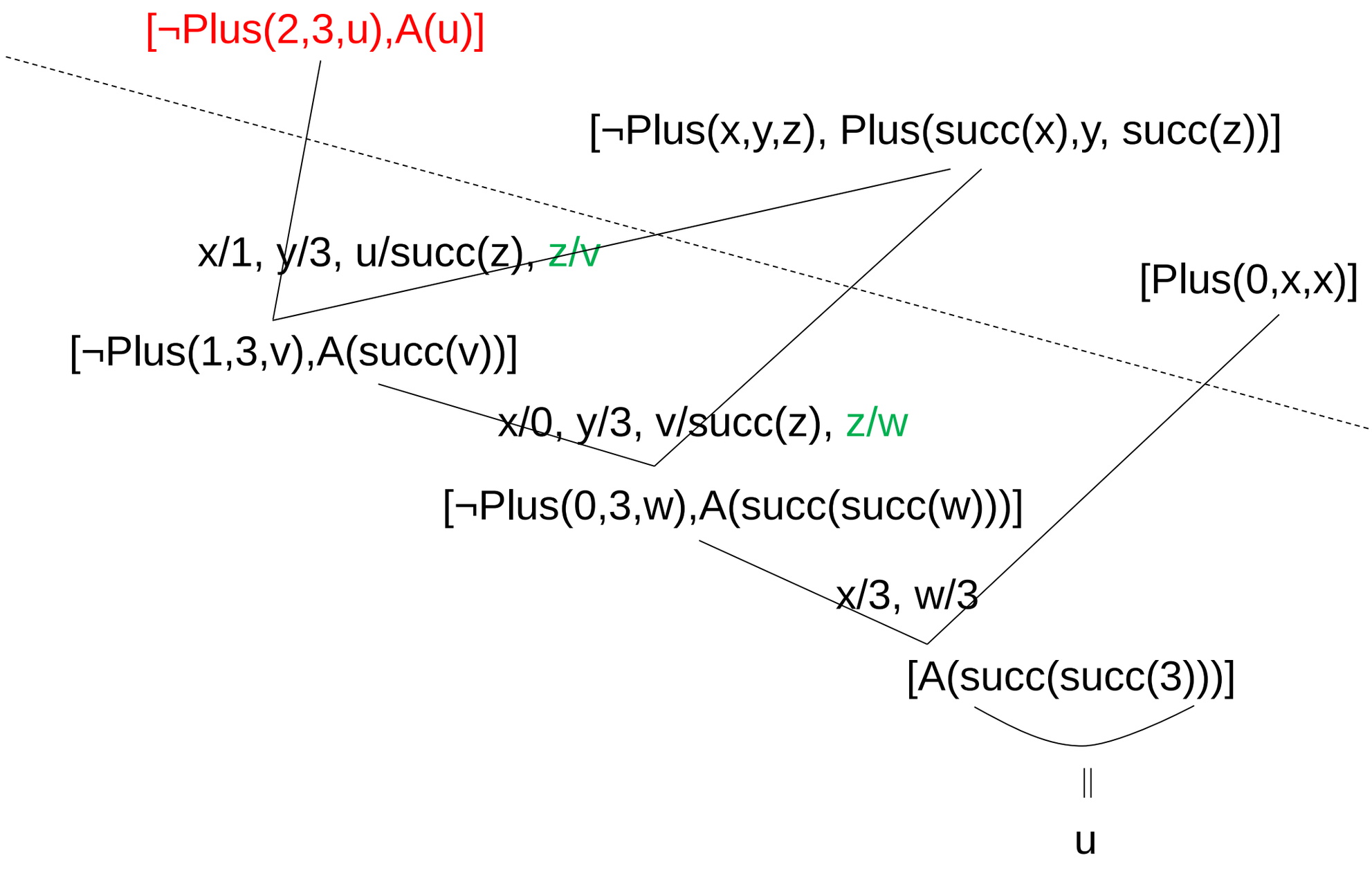
Since A does not appear anywhere else, it will not be possible to derive the empty clause. Therefore, the derivation ends when we produce a clause containing only the answer predicate.

Example 1

KB
- ∀x.Plus(zero,x,x)
- ∀x∀y∀z.Plus(x,y,z))⊃Plus(succ(x),y,succ(z))

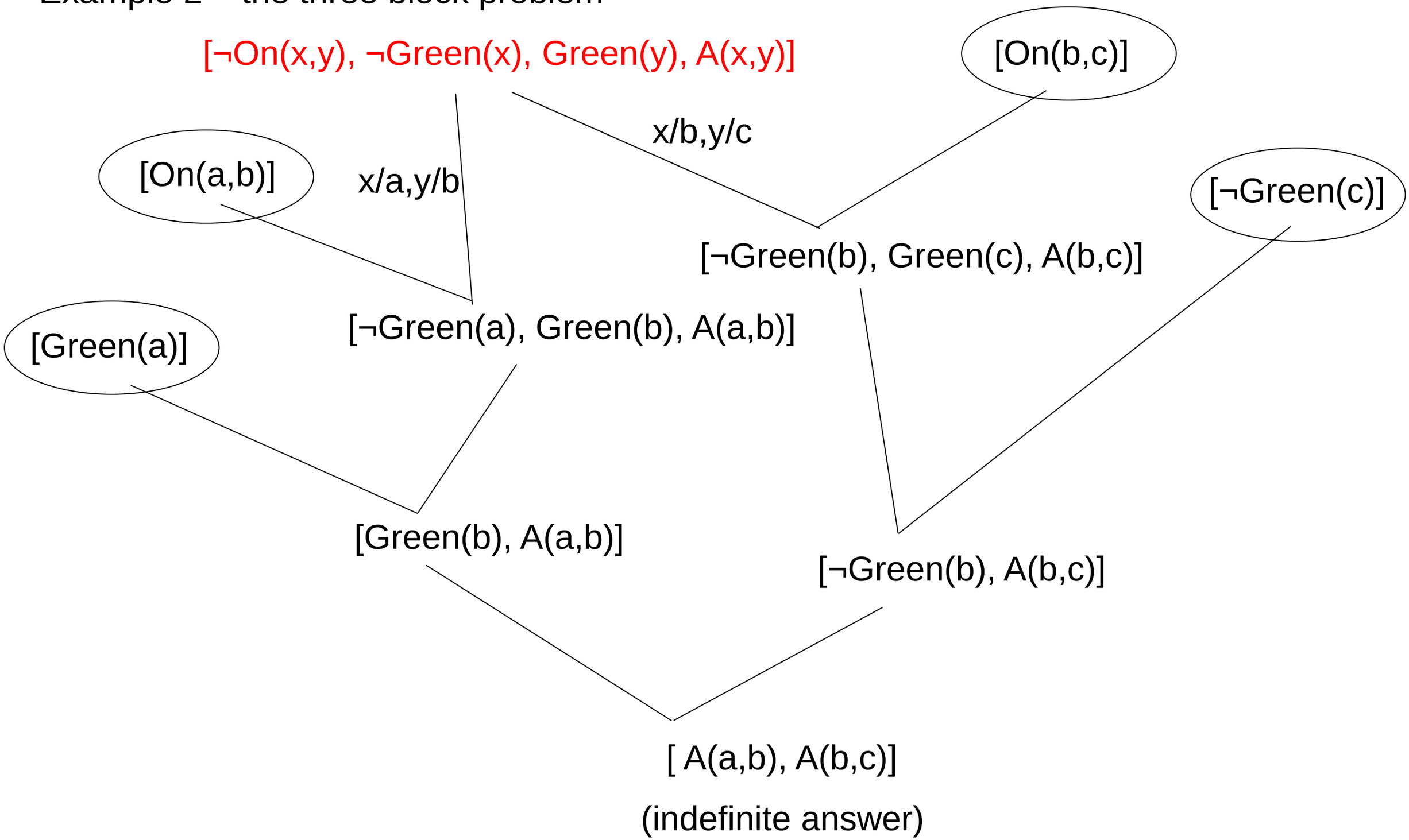
Question: ∃u.Plus(2,3,u) (we add ∧¬A(x))

Resolution – answer extraction



Resolution – answer extraction

Example 2 – the three block problem



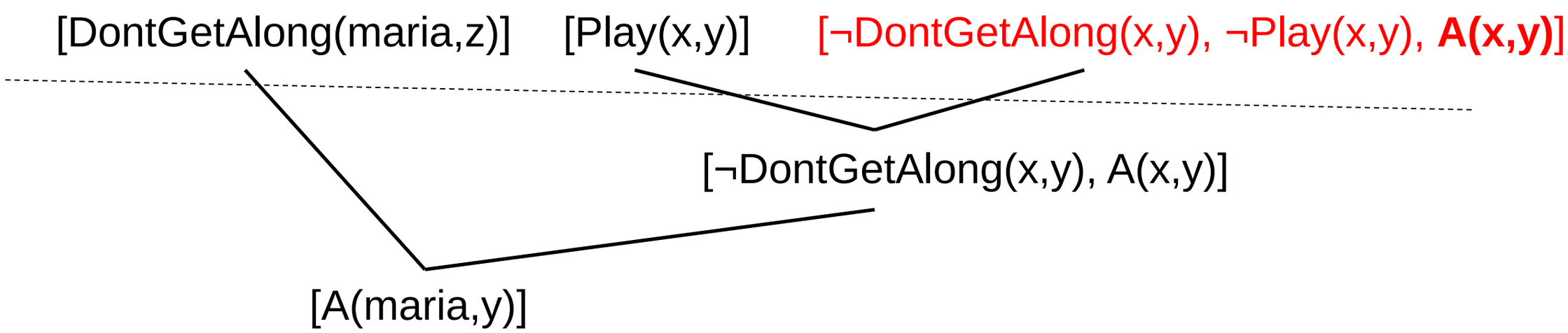
Resolution – answer extraction

Obs. We can get answers containing variables.

For example

KB
- ∀z.DontGetAlong(maria,z)
- ∀x∀y.Play(x,y)

Question ∃x∃y.DontGetAlong(x,y) ^Play(x,y)



We get a derivation with the final clause [A(maria,y)] interpreted as “the answer is any instance of the terms (maria, y)”.

Skolemization

It is the process of removing existential quantifiers by elimination.

Idea: we introduce unique names for each variable quantified with an existential.

For example, for

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

we name x a
z f(y)

➔ we will use instead $\forall y . P(a,y,f(y))$

a and f are called Skolem symbols (they do not appear anywhere else).

See also Ch. 9.5 - Resolution in Stuart J. Russell, Peter Norvig. Artificial Intelligence – a modern approach, 3rd edition, Pearson, 2010.

Skolemization

In general, Skolemization replace each existential variable by a new function symbol with as many arguments as there are universal variable dominating the existential.

$\alpha \qquad \forall x_1(\ldots\forall x_2(\ldots\forall x_3(\ldots\exists y[\ldots y\ldots]\ldots)\ldots)\ldots)$

Skolemization

$\alpha' \qquad \forall x_1(\ldots\forall x_2(\ldots\forall x_3(\ldots[\ldots f(x_1,x_2,x_3)\ldots]\ldots)\ldots)\ldots)$

where f appears nowhere else.

Obs. $\not\models (\alpha \equiv \alpha')$

For example, $\exists x.kill(x,victim)$ strictly speaking is not logically equivalent to kill(murderer,victim).

Skolemization

Att. Do not confuse these substitutions (i.e., Skolem constants) with the interpretations used to define the semantics of quantifiers.

The substitution replaces a variable with a term (it's syntax) to produce new sentences, whereas an interpretation maps a variable to an object in the domain.

Prop. α and α′ are inferentially equivalent, that is
KB U {α} is satisfiable iff KB U {α′} is satisfiable.

Skolemization

Att: For logical correctness, it is important to have the dependence of variables right.

For example, $\exists x \forall y.R(x,y) \vDash \forall y \exists x.R(x,y)$ but the converse does not hold

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

↓ CNF

$\{R(a,y), \neg\, R(x,b)\}$ a,b Skolem constants

↓ x/a, y/b

$\square$

Skolemization

Att: For logical correctness, it is important to have the dependence of variables right.

For example, $\exists x \forall y.R(x,y) \vDash \forall y \exists x.R(x,y)$ but the converse does not hold

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

↓ CNF

$\{R(a,y), \neg\ R(x,b)\}$ a,b Skolem constants

↓ x/a, y/b

[]

The converse:

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

↓ CNF

$\{R(f(y),y), \neg\ R(x,g(x))\}$ f,g Skolem functions

do not unify

Skolemization

Example 3 [taken from https://www.cs.utexas.edu/users/novak/reso.html]

KB
- All hounds howl at night.
- Anyone who has any cat will have no mice.
- Light sleepers do not have anything that howls at night.
- John has either a cat or a hound.

Question: If John is a light sleeper, then John does not have any mice.

Skolemization

Example 3 [taken from https://www.cs.utexas.edu/users/novak/reso.html]

KB
- All hounds howl at night.
- Anyone who has any cat will have no mice.
- Light sleepers do not have anything that howls at night.
- John has either a cat or a hound.

Question: If John is a light sleeper, then John does not have any mice.

$\forall x.Hound(x) \supset Howl(x)$

$\forall x \forall y \forall z.((Have(x,y) \wedge Cat(y)) \supset \neg(Have(x,z) \wedge Mouse(z)))$

$\forall x \forall y(Ls(x) \supset \neg(Have(x,y) \wedge Howl(y)))$

$\exists x.((Have(john,x) \wedge (Cat(x) \vee Hound(x)))$

Question: $\forall x(Ls(john) \supset \neg(Have(john,x) \wedge Mouse(x)))$

Skolemization

∀x.Hound(x)⊃Howl(x)

∀x∀y∀z.((Have(x,y)^Cat(y))⊃ ¬(Have(x,z)^Mouse(z)))

∀x∀y(Ls(x)⊃ ¬(Have(x,y)^Howl(y)))

∃x.((Have(john,x) ^ (Cat(x)vHound(x)))

∃x¬(Ls(john)⊃ ¬(Have(john,x)^Mouse(x)))

CNF

[¬Hound(x),Howl(x)]

[¬Have(x,y), ¬Cat(y), ¬Have(x,z), ¬Mouse(z)]

[¬Ls(x), ¬Have(x,y), ¬Howl(y)]

[Have(john,a)]

[Cat(a),Hound(a)] a Skolem constant

[Ls(john)]

[Have(john,b)]

[Mouse(b) b Skolem constant

apply resolution ➔ [ ]

Equality

If we treated = as a predicate, we would miss, for example, that

{a=b, b=c, a≠c} is unsatisfiable.

For this reason, it is necessary to add the clausal versions of the axioms of equality:

- reflexivity $\forall x.x=x$
- symmetry $\forall x \forall y.x=y \supset y=x$
- tranzitivity $\forall x \forall y \forall z.x=y \wedge y=z \supset x=z$
- substitution for functions

  $\forall x_1 \forall y_1 \ldots \forall x_n \forall y_n.x_1=y_1 \wedge \ldots \wedge x_n=y_n \supset f(x_1,\ldots,x_n)= f(y_1,\ldots,y_n)$ for every function symbol f of arity n
- substitution for predicates

  $\forall x_1 \forall y_1 \ldots \forall x_n \forall y_n.x_1=y_1 \wedge \ldots \wedge x_n=y_n \supset P(x_1,\ldots,x_n) \equiv P(y_1,\ldots,y_n)$ for every predicate symbol P of arity n

Now = can be treated as a binary predicate.

Equality

Example 4

KB
$$\begin{bmatrix} \forall x.\text{Married}(\text{father}(x),\text{mother}(x)) \\ \text{father}(\text{john})=\text{bill} \end{bmatrix}$$

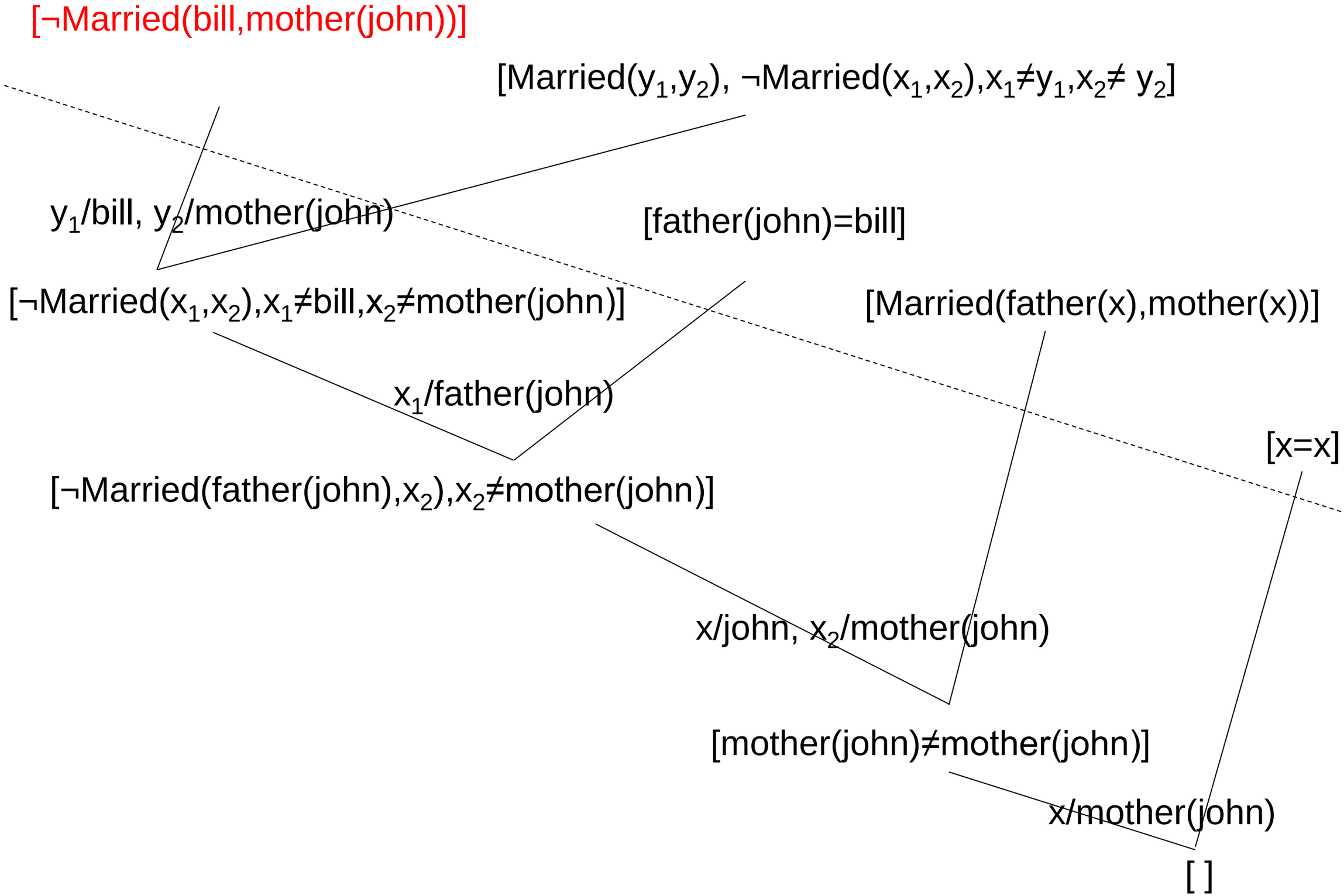
Question: Married(bill,mother(john))

$$\forall x_1 \forall y_1 \forall x_2 \forall y_2 . x_1 = y_1 \wedge x_2 = y_2 \supset \text{Married}(x_1,x_2) \equiv \text{Married}(y_1,y_2)$$

↓ CNF (replace ⊃ and ≡; distribute ∧ over ∨; collect terms)

$$[\text{Married}(y_1,y_2),\ \neg\text{Married}(x_1,x_2), x_1 \neq y_1,\ x_2 \neq y_2]$$

Equality


[¬Married(bill,mother(john))]
[Married(y1,y2), ¬Married(x1,x2),x1≠y1,x2≠ y2]
y1/bill, y2/mother(john)
[father(john)=bill]
[¬Married(x1,x2),x1≠bill,x2≠mother(john)]
[Married(father(x),mother(x))]
x1/father(john)
[x=x]
[¬Married(father(john),x2),x2≠mother(john)]
x/john, x2/mother(john)
[mother(john)≠mother(john)]
x/mother(john)
[ ]

Resolution – computational intractability

Resolution does not provide a general effective solution for automated reasoning.

The FOL case

KB: ∀x∀y.LessThan(succ(x),y)⊃LessThan(x,y)

Question: LessThan(zero,zero)

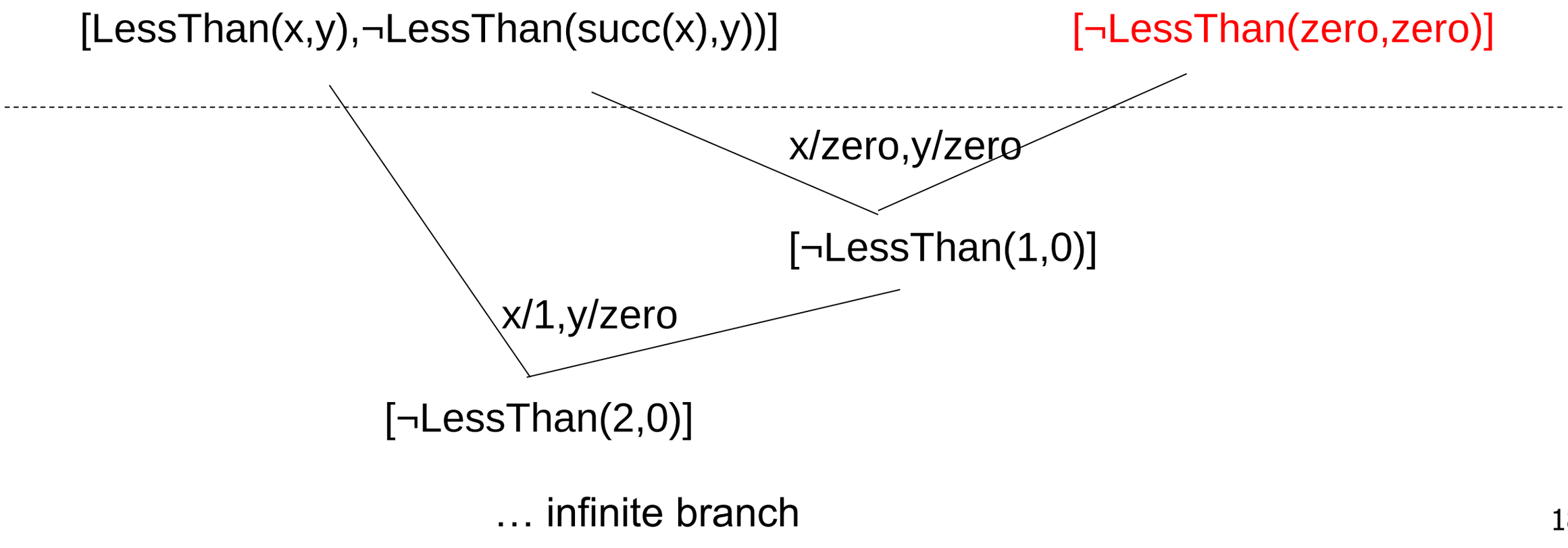
Resolution – computational intractability

Resolution does not provide a general effective solution for automated reasoning.

The FOL case

KB: ∀x∀y.LessThan(succ(x),y)⊃LessThan(x,y)

Question: LessThan(zero,zero)



Resolution – computational intractability

If we apply a depth-first procedure to search for the empty clause, we can go on an infinite branch.

We cannot detect whether a branch will continue indefinitely.

But we know that S ⊨ [] iff S ⊢ [], meaning that if a set of sentences is unsatisfiable, then there is a branch in the derivation containing the empty clause.

So, a depth-first search would guarantee to find it (the unsatisfiable case). But when the clauses are satisfiable, the search may or may not end.

Resolution – the Herbrand theorem

In the propositional case, the resolution procedure always terminates (in the initial set of clauses there is a finite number of literals).

In some cases, Resolution in FOL reduces to the propositional case.

Def. Given S a set of clauses, the Herbrand universe of S, written H_S, is the set of all ground terms (i.e. terms with no variables) formed using just the constants and the function symbols in S (if S has no constants or function symbols, we use just a constant a).

For example, if

$S=\{[\neg P(x,f(x,a)), \neg Q(x,a),R(x,b)]\}$

then

$H_S=\{a,b,f(a,a),f(a,b),f(b,a),f(b,b),f(a,f(a,a)),\ldots\}$

Resolution – the Herbrand theorem

Def. The Herbrand base of S, written $H_S(S)$, is the set of all ground clauses cθ, where c∈S and θ assigns the variables in c to terms in the Herbrand univers.

For the same S as before, we have:

$H_S(S)$={[¬P(a,f(a,a)), ¬Q(a,a),R(a,b)],

[¬P(b,f(b,a)), ¬Q(b,a),R(b,b)],

[¬P(f(a,a),f(f(a,a),a)), ¬Q(f(a,a),a),R(f(a,a),b)],

[¬P(f(b,a),f(f(b,a),a)), ¬Q(f(b,a),a),R(f(b,a),b)],…}

Resolution – the Herbrand theorem

Herbrand's Theorem. A set of clauses is satisfiable iff its Herbrand base is satisfiable.

This is important because the Herbrand base is a set of clauses without variables, so it is essentially propositional.

The difficulty is that typically the Herbrand base is an infinite set of propositional clauses (but finite when the Herbrand universe is finite – no function symbols and a finite number of constants in S).

Sometimes, the Herbrand universe can be kept finite by looking at the type of the arguments and values of functions, and including terms like f(t) only if the type of t is appropriate for f (e.g. we may exclude terms like birthday(birthday(john)) from the Herbrand universe).

Resolution – complexity

The propositional case

How long may it take for the resolution procedure on a finite set of propositional clauses to terminate?

In 1985, Armin Haken proved that there exist unsatisfiable propositional clauses $c_1,\ldots,c_n$ so that the shortest derivation of the empty clause has the length of order 2^n.

Resolution takes an exponential time, no matter how well we choose the derivations.

Is there a better way to determine whether a set of propositional clauses is satisfiable? – one of the most difficult questions in computer science.

Resolution – complexity

In 1972, Stephen Cook proved that the satisfiability problem was NP-complete.

Any search problem (e.g. scheduling, routing) where we are searching for an element that satisfies a certain property, and where we can test in polynomial time whether a candidate satisfies the property, can be converted into a propositional satisfiability problem.

Thus, a polynomial time algorithm for satisfiability would imply a polynomial time for all these search problems.

Resolution

We may need to consider alternative options for resolution:

- Procedural representations – give more control over the reasoning process to the user
- Using representation languages that are less expressive than FOL – e.g. description languages.

The research in KRR approaches both directions.

In some applications it may be worth waiting (for a long time) for answers.

There is an area of AI called automated theorem-proving, that uses resolution (among other procedures) for this purpose (e.g. to determine whether Goldbach's Conjecture follows from the axioms of number theory).

SAT solvers

They are procedure that determine the satisfiability of a set of clauses more efficiently than the resolution.

They search for an interpretation that would prove the clauses to be satisfiable.

They are often applied to clauses that are known to be satisfiable, but the satisfying interpretation in not known.

SAT solvers

They are procedure that determine the satisfiability of a set of clauses more efficiently than the resolution.

They search for an interpretation that would prove the clauses to be satisfiable.

They are often applied to clauses that are known to be satisfiable, but the satisfying interpretation is not known.

When C is a set of clauses and m is a literal, C•m is defined as following:

$$C\bullet m=\{c \mid c\in C,\ m\notin c,\ \bar{m}\notin c\} \cup \{(c-\bar{m} \mid c\in C,\ m\notin c,\ \bar{m}\in c\}$$

For example, if C={[p,q],[$\bar{p}$,a,b],[$\bar{p}$,c],[d,e]} then

C•p={[a,b],[c],[d,e]}

C•$\bar{p}$={[q],[d,e]}

(C•$\bar{p}$) •$\bar{q}$={[],[d,e]}

(C•$\bar{p}$) •q={[d,e]}

((C•$\bar{p}$) •q) •d={}

SAT solvers

Given an interpretation $\mathcal{I}$,

- If p is true then C is satisfiable iff C•p is satisfiable
- If p is false then C is satisfiable iff C•$\bar{p}$ is satisfiable

Procedure DP (Davis-Putnam)

Input: a set of clauses C

Output: are the clauses satisfiable YES or NO

```
procedure DP(C)
if (C is empty) then return YES
if (C contains [ ]) then return NO
let p be some atom in C
if (DP(C•p)=YES) then return YES
                 else return DP(C•p̄)
```

SAT solvers

Strategies for choosing an atom p:

- p appears in the most clauses in C;
- p appears in the fewest clauses in C;
- p is the most balanced atom in C (i.e. the number of positive occurrences in C is closest to the number of negative occurrences);
- p is the least balanced atom;
- p appears in the shortest clause in C;

For the propositional case, the DP procedure is the fastest one in practice, among all SAT solvers (still, for some inputs it takes exponential time!). Thus, problems with tens of millions of variables can be approached.

SAT solvers revolutionized fields such as hardware verification or security protocols verifications.

SAT solver - example

KB
- Toddler
- Toddler⊃Child
- Child∧Male⊃Boy
- Infant⊃Child
- Child∧Female⊃Girl
- Female

Question: Girl

KB U {¬α}

Input: [Toddler], [¬Toddler,Child], [¬Child,¬Male,Boy], [¬Infant,Child],
[¬Child,¬Female,Girl], [Female], [¬Girl]

Output: false (unsat)

SAT solver - example

Input: [Toddler], [¬Toddler,Child], [¬Child,¬Male,Boy], [¬Infant,Child],
[¬Child,¬Female,Girl], [Female], [Girl]

DP procedure

Output: [Toddler/true, Child/true, n(Male)/true, n(Female)/false, Girl/true]

Obs. The solution is not unique.

Obs. It is not necessary that all the propositional symbols have a truth value – see Infant and Boy.