Resolution

Introduction to Artificial Intelligence

G. Lakemeyer

Winter Term 2016/17

Deduction at the Knowledge Level

What should a deductive inference method compute?

Given a KB and α, determine whether $\mathrm{KB} \models \alpha$ holds, or, given an open wff $\alpha(x_1, x_2, \ldots, x_n)$, find $t_1, t_2, \ldots, t_n$ such that

$$\mathrm{KB} \models \alpha(t_1, t_2, \ldots, t_n).$$

(t_i are closed terms)

A KB is usually finite, i.e., $\mathrm{KB} = \{\alpha_1, \ldots, \alpha_k\}$. Hence,

$$\begin{aligned} \mathrm{KB} \models \alpha \text{ iff } & \models [(\alpha_1 \wedge \ldots \wedge \alpha_k) \supset \alpha] \\ \text{iff } & \mathrm{KB} \cup \{\neg\alpha\} \text{ is unsatisfiable} \\ \text{iff } & \mathrm{KB} \cup \{\neg\alpha\} \models \text{FALSE} \end{aligned}$$

Hence we are looking for a mechanism which either tests for validity, satisfiability, or checks whether FALSE can be inferred.

We will now look at just such a method, using a language very close to FOL (ignoring quantifiers for now).

Clausal Form

Formulas:	are sets of clauses.
Clauses:	are sets of literals.
Literals:	are atomic sentences or their negation (positive or negative literals).

Notation:

- If l is a literal, then $\sim l$ is its complement:

$$\sim p \Rightarrow \neg p, \qquad \sim(\neg p) \Rightarrow p$$

- To distinguish clauses from formulas:
  - use [and] for clauses: $[p, \sim r, s]$
  - use { and } for formulas: $\{[p, \sim r, s], [p, r, s], [\sim p]\}$
  - [ ] is the empty clause, {} the empty formula.

Interpreting clauses as FOL-formulas

A formula is understood as a conjunction of clauses.

A clause is understood as a disjunction of literals.

Literals have their usual meaning.

Hence,

- $\{[p, \sim q], [r], [s]\}$ represents $((p \vee \neg q) \wedge r \wedge s)$.
- $[\,]$ represents FALSE.
- $\{\}$ represents TRUE.

In general, every formula (in the new sense) represents a wff in conjunctive normal form.

CNF and DNF

Tranformation to CNF

Every propositional wff α can be converted to an α' in conjunctive normal form (CNF) such that $\models \alpha \equiv \alpha'$.

1. Eliminate $\supset$ and $\equiv$ with $\alpha \supset \beta \;\Rightarrow\; (\neg\alpha \vee \beta)$ etc.
2. Push $\neg$ "inwards" with $\neg(\alpha \wedge \beta) \;\Rightarrow\; (\neg\alpha \vee \neg\beta)$ etc.
3. Distribute $\vee$ over $\wedge$ with $((\alpha \wedge \beta) \vee \gamma) \;\Rightarrow\; ((\alpha \vee \gamma) \wedge (\beta \vee \gamma))$.
4. Simplify: $(\alpha \vee \alpha) \;\Rightarrow\; \alpha$, $\neg\neg\alpha \Rightarrow\; \alpha$, etc.

The result is a conjunction of disjunctions of literals.

Similarly, any wff can be transformed into an equivalent disjunctive normal form (DNF).

CNF and Clausal Form

CNF wffs can be identified with clausal forms:

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

Hence: given a finite KB and α, in order to test whether $\mathrm{KB} \models \alpha$ holds, it suffices to do the following:

1. Transform $(\mathrm{KB} \wedge \neg\alpha)$ into CNF.
2. Test whether this CNF is satisfiable.

The Inference Rule of Resolution

Resolvent

Given two clauses, infer a new clause:

From $\{p\} \cup C_1$ and $\{\sim p\} \cup C_2$ infer $C_1 \cup C_2$.

$C_1 \cup C_2$ is called the resolvent of the input clauses relative to p.

Example:

The resolvent of $[w, p, q]$ and $[w, s, \sim p]$ relative to p is $[w, q, s]$.

Special case:

$[p]$ and $[\sim p]$ resolve to $[\,]$. (C_1 and C_2 are empty.)

Derivation

A derivation of a clause c from a set of clauses S is a sequence $c_1, c_2, \ldots, c_n$ of clauses, where $c_n = c$ and for all c_i,

1. $c_i \in S$ or
2. c_i is a resolvent of c_j and c_k with $j, k < i$.

We write $S \longrightarrow c$ for the derivation of c from S.

Why is Resolution Ok?

While resolution is an inference rule at the symbol level, there is a simple connection to logical interpretations at the knowledge level.

Resolvents are implications of the input clauses.

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

Case 1: Let $I \models p$. Then $I \models \beta$ and hence $I \models (\alpha \vee \beta)$.

Case 2: Let $I \not\models p$ Then $I \models \alpha$ and hence $I \models (\alpha \vee \beta)$.

Therefore, in any case, $I \models (\alpha \vee \beta)$. Thus $\{(p \vee \alpha), (\neg p \vee \beta)\} \models (\alpha \vee \beta)$.

Special case:

$[p]$ and $[\sim p]$ resolve to $[\,]$.
Hence $\{[p], [\sim p]\} \models \text{FALSE}$, or, $\{[p], [\sim p]\}$ is unsatisfiable.

Derivation and Implication (1)

Theorem:

The previous theorem can be generalized to derivations:

$$\text{If } S \longrightarrow c \text{ then } S \models c.$$

Proof idea: Induction over the length of a derivation. Show, using case analysis as above, that $S \models c$.

Note:

The converse does **not** hold, that is, sometimes $S \models c$ holds, yet $S \longrightarrow c$ does not hold.

Example: $\{[\sim p]\} \models [\sim p, \sim q]$, i.e. $\neg p \models (\neg p \vee \neg q)$. But there is no derivation!

Derivation and Implication (2)

On the other hand. . .

Theorem:

Resolution is correct and complete for [ ].

$$S \longrightarrow [\,]\text{ iff } S \models \text{FALSE}.$$

[Theorem can be generalized to formulas with quantifiers (see later)].

Hence we have for arbitrary sets of clauses S:

S is unsatisfiable iff $S \longrightarrow [\,]$.

Can be turned into a method to test for unsatisfiability (and, therefore, implication):
Search through the space of derivations and check if [ ] obtains.

Computing implications

To determine whether $\mathrm{KB} \models \alpha$ holds:

- transform KB and $\neg\alpha$ into CNF (results in S);
- test whether $S \longrightarrow [\,]$ (If $\mathrm{KB} = \{\}$, then test if α is valid.)

Non-deterministic method:

1. Test if $[\,]$ in S is.
   If so, then return *unsatisfiable*.
2. Test if there are clauses c_1, c_2 in S resolving to c_3 where $c_3 \notin S$.
   If not, then return *satisfiable*.
3. Add c_3 to S and goto 1.

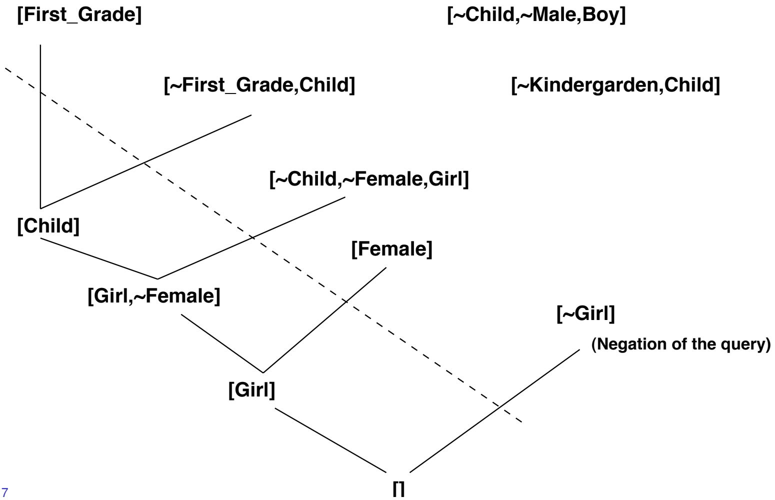
Note: KB needs to be converted to CNF only once.

- Handles multiple queries with the same KB.
- If KB is extended by α, then only the CNF of α needs to be computed.

Example 1

$\mathrm{KB} = \{$First_Grade, First_Grade $\supset$ Child, Child $\wedge$ Male $\supset$ Boy,
Kindergarden $\supset$ Child, Child $\wedge$ Female $\supset$ Girl, Female$\}$

Show: $\mathrm{KB} \models$ Girl



Quantifiers

Clauses as before, but atoms are $P(t_1, \ldots, t_n)$, where t_i may contain variables. (Equality atoms $(t_i = t_j)$ are excluded.)

Interpretations as wffs as before, but variables are implicitly understood to be universally quantified.

Example: $\{[P(x), \sim R(a, f(b, x))], [Q(x, y)]\}$
stands for $\forall x \forall y \{[R(a, f(b, x)) \supset P(x)] \wedge Q(x, y)\}$.

Substitutions: $\theta = \{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$.

Notation: If l is a literal and θ a substitution, then $l\theta$ is the result of the substitution. (Analogously we define $c\theta$ for clauses c.)

Ex.: $\theta = \{x/a, y/g(x, b, z)\}$: $P(x, z, f(x, y))\theta = P(a, z, f(a, g(x, b, z)))$.

A ground literal is a literal without variables.
A literal l is an instance of l' if there is a θ such that $l = l'\theta$.

Generalizing CNF

To generalize resolution, we first show how to convert FOL wffs into CNF.

FOL $\rightarrow$ CNF

1. Eliminate $\supset$ and $\equiv$.
2. Push $\neg$ inside. ($\neg\forall x\alpha \;\Rightarrow\; \exists x\neg\alpha$ etc.)
3. Rename variables to make them syntactically distinct.
   (E.g. $\exists x[P(x)] \wedge Q(x) \;\Rightarrow\; \exists z[P(z)] \wedge Q(x)$, z a new variable)
4. Eliminate $\exists$'s (deferred to later).
5. Move $\forall$'s to the left.
   (exampleE.g. $\alpha \wedge \forall x\beta \;\Rightarrow\; \forall x[\alpha \wedge \beta]$, where α does not contain x.)
6. Distriute $\vee$ over $\wedge$.
7. Simplify.

... results in quantified conjunctions of disjunctions.

To obtain clausal form simply eliminate all $\forall$'s.

Clauses with Variables

Main idea:

A literal with variables represents all its instances. We allow inference over all instances.

Hence, given

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

we would like to infer

$[\sim Q(b), \sim R(b,f(a))]$

because

$[P(b,a), \sim Q(b)]$ is an instance of $[P(x,a), \sim Q(x)]$ and
$[\sim P(b,a), \sim R(b,f(a))]$ of $[\sim P(b,y), \sim R(b,f(y))]$.

First-Order Resolution

Given: clauses $\{l_1\} \cup C_1$ and $\{\sim l_2\} \cup C_2$.

- Rename the variables so that they are different in both clauses.
- For every θ with $l_1\theta = l_2\theta$ we can infer $(C_1 \cup C_2)\theta$.

We say that l_1 and l_2 are unifiable or that θ is a unifier of the literals.

Derivations are defined as before. Then,

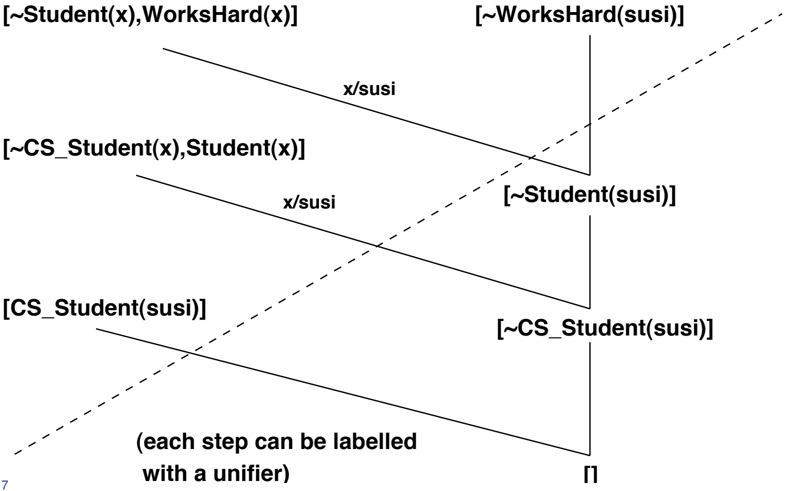
Theorem

$$S \longrightarrow [\,]\ \text{iff}\ S \models \text{FALSE}.$$

Example 2

KB$= \{\forall x\; CS_Student(x) \supset Student(x), \forall x\; Student(x) \supset WorksHard(x),$
$CS\_Student(susi)\}$

Question: $WorksHard(susi)$



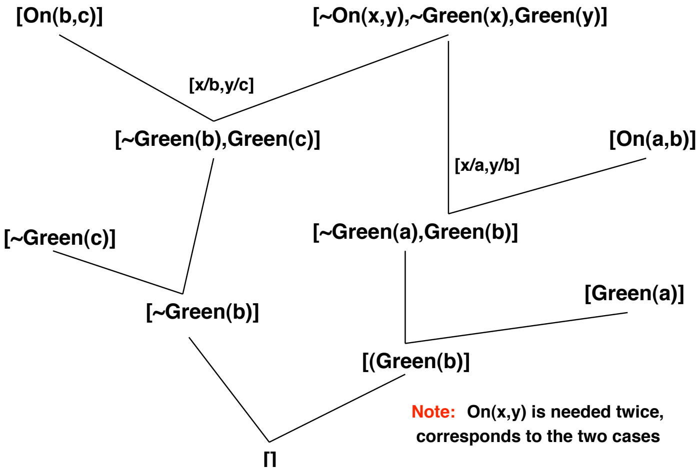
(each step can be labelled with a unifier)

The 3-Blocks Example Revisited

$\mathrm{KB} = \{On(a,b), On(b,c), Green(a), \neg Green(c)\}$ (is in CNF!)

$F = \exists x \exists y [On(x,y) \wedge Green(x) \wedge \neg Green(y)]$

Note: there are no $\exists$ to eliminate in $\neg$F,
results in $\{[\sim On(x,y), \sim Green(x), Green(y)]\}$ in CNF.

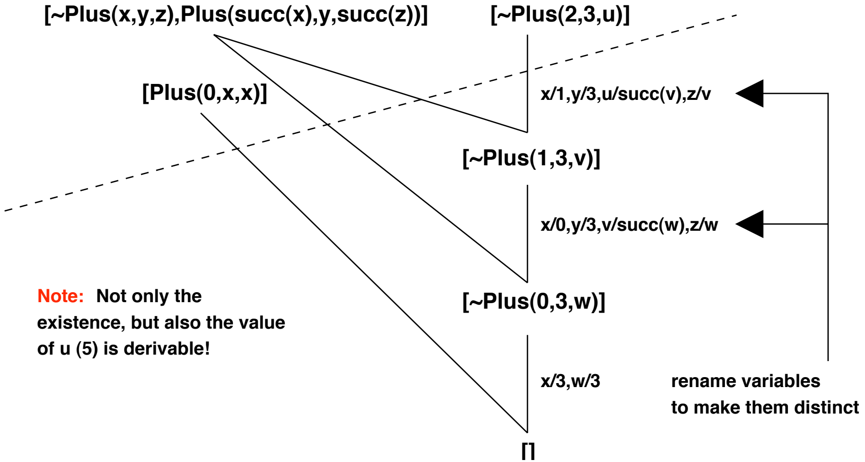


Note: On(x,y) is needed twice, corresponds to the two cases

Arithmetic

KB$= \{\forall x\ Plus(null, x, x), \forall x \forall y \forall z\ Plus(x, y, z) \supset Plus(succ(x), y, succ(z))\}$

$F = \exists u Plus(2, 3, u)$ (here 0 stands for *null*, 1 for $succ(null)$, etc.)



Note: Not only the existence, but also the value of u (5) is derivable!

Answer Extraction

In FOL it is possible to infer $\exists x P(x)$ without inferring $P(t)$ for any t.

e.g. in the 3-blocks problem:
$\exists x \exists y [On(x, y) \land Green(x) \land \neg Green(y)]$ follows,
yet we cannot say which block it is.

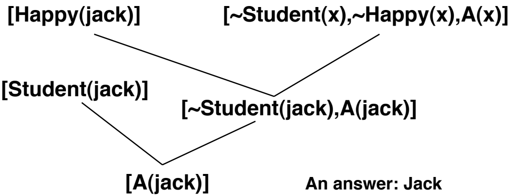
Solution: Answer-predicates

- Replace the query $\exists x P(x)$ by $\exists x [P(x) \land \neg A(x)]$, where A (the answer predicate) occurs nowhere else.
- Instead of inferring [ ] infer a clause which contains only the answer predicate.
- Still results in a sound and complete inference method.

Example Using an Answer Predicate

KB= $\{Student(jack), Student(susi), Happy(jack)\}$

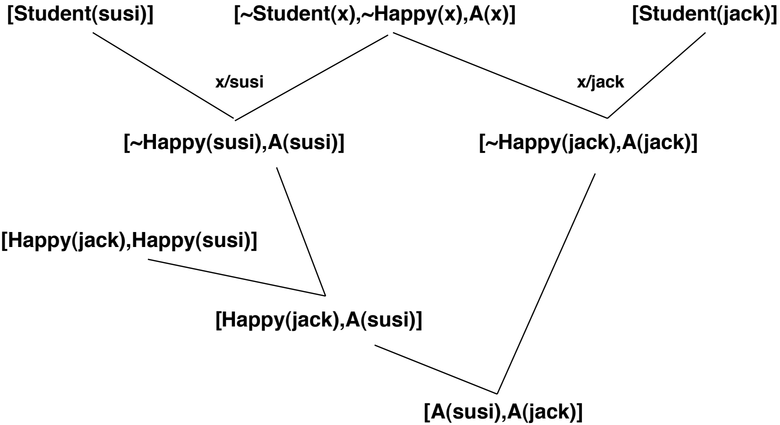
Query: $\exists x[Student(x) \land Happy(x)]$



Disjunctive Answers

Example: $\mathrm{KB}= \{Student(jack), Student(susi), Happy(jack) \vee Happy(susi)\}$

Query: $\exists x[Student(x) \wedge Happy(x)]$



Note: Variables may appear in the answer.

Skolemization

So far we ignored $\exists$ when converting into CNF.

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

Idea: Make up names for those individuals. These are called Skolem-constants and -functions.

There is an x: call it a.

For all y there is an z: call them $f(y)$.

Then we get $\forall y P(a, y, f(y))$.

In general:

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

is replaced by

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

where f is a function symbol occurring nowhere else.

Why Skolemization is Ok

Skolemizations is not equivalence preserving.

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

But satisfiability is retained:

Theorem

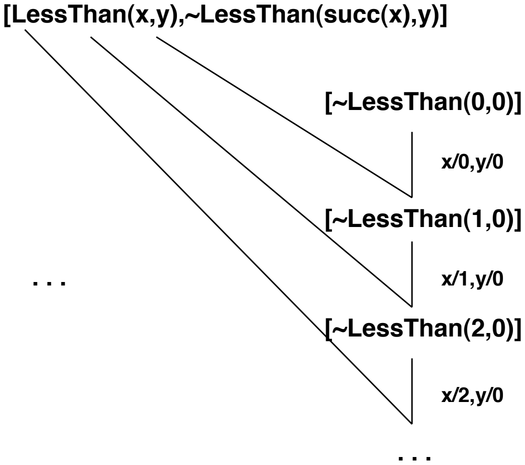
α is satisfiable iff α' is satisfiable, where α' is the result of Skolemization.

That is sufficient for resolution!

A Problem

KB$= \{\forall x \forall y \mathit{LessThan}(\mathit{succ}(x), y) \supset \mathit{LessThan}(x, y)\}$

$F = \mathit{LessThan}(\mathit{null}, \mathit{null})$ should fail because KB$\not\models F$.



An infinite branch of resolvents! Simple DFS won't work to derive [ ].

Undecidability

Is it possible in general to detect infinite branches?

No! FOL is too powerful.

FOL is as expressive as Turing machines.

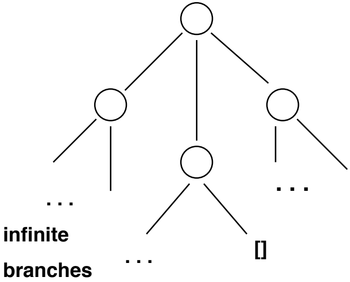
The above problem is as hard as the Halting Problem.

There is no function which does the following:

Function FOL-Sat [clauses] $=$
  If clauses are unsatisfiable
    then return YES
    else return NO.

Resolution is Complete

If the set of clauses is unsatisfiable, then there is a branch with [ ] as a leaf node.



Hence Breadth-First Search guarantees that [ ] will be found if derivable.

Search does not necessarily terminate if the clauses are satisfiable.

$\Rightarrow$ Satisfiability in FOL is semi-decidable.

Too Specific Unifiers

Termination or even efficiency cannot be guaranteed in general,

... but there is room for improvement.

One possibility:

Reduce search redundancy by keeping the search as general as possible.

Example:

$[\ldots, P(g(x), f(x), z)] \quad [\sim P(y, f(w), a), \ldots]$ is unifiable with

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

but also with

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

Sometimes [ ] is not derivable because of too specific substitutions.

Most General Unifier

MGU

θ is a most general unifier (MGU) of the literals l_1 and l_2 iff

1. θ unifies l_1 and l_2
2. for every unifier θ' there is a substitution θ^* such that $\theta' = \theta\theta^*$

Example:

In the previous example the MGU is
$\theta = \{x/w, y/g(w), z/a\}$
and therefore
$\theta_1 = \theta\{w/b\}$
$\theta_2 = \theta\{w/f(z)\}$

Theorem

Resolution is complete when restricted to MGUs.

Computing the MGU

Computing the MGU for a set of literals l_i:

1. Start with $\theta = \{\}$.
2. If all $l_i\theta$ are identical, then success: θ is the MGU.
   Otherwise look for the disagreement set DS,
   e.g. $P(a, f(a, g(z), \ldots \quad P(a, f(a, u, \ldots$
   $DS = \{u, g(z)\}$.
3. Find a variable $v \in DS$ and a term $t \in DS$ which does not contain v (occur check). Otherwise abort: not unifiable.
4. $\theta = \theta\{v/t\}$
5. Goto 2.

Note:

There are faster linear MGU-algorithms.

Herbrand Theorem

Sometimes FOL theories (sets of FOL sentences) can be transformed into propositional sentences.

Let S be a set of clauses.

- The Herbrand universe of S is the set of all terms which can be formed from the the functions and terms in S.

  e.g.: if S contains the unary function f and constants c and d, then $U = \{c, d, f(c), f(d), f(f(c)), f(f(d)), f(f(f(c))), \ldots\}$.
- the Herbrand base of S is $\{c\theta \mid c \in S \text{ and } \theta \text{ replaces variables by terms from the Herbrand univ.}\}$.

Theorem

S is satisfiable iff the Herbrand base is satisfiable.

When is This Useful?

Herbrand bases have no variables and are therefore propositional, but they are generally infinite.

- finite if the Herbrand universe is finite;
- sometimes type restrictions can be used to obtain a finite Herbrand basis.
  $\Rightarrow$ use $f(t)$ only if t is of the right type.

Resolution is Hard!

First-order resolution does not always terminate.

What about the propositional case?

Armin Haken showed that there are clauses $\{c_1, \ldots, c_n\}$ such that the shortest derivation of [ ] contains 2^n clauses.

Methods based on Resolution run in exponential time (in the worst case).

Is that a problem only of resolution?

Probably not!

Testing satisfiability of propositional clauses (SAT) is NP-complete [Cook 1971].

While not proven it seems unlikely that NP-complete problems can be solved in polynomial time.

What Does That Mean for KR?

Problem: The entailments of a KB are usually only useful when they can be derived fast.

Entailment in FOL is probably too hard for KR.

What to do?

- Leave search control to the user.
- Use less expressive languages (examplee.g. Horn clauses)

Note: Sometimes it's ok to wait a long time (e.g. when proving mathematical theorems).

In any case, it is important to avoid redundancies, or in general, to tune resolution in order to shorten the search.

Strategies for Resolution

1. Eliminating clauses

- Pure clauses

  A clause contains a literal l, and $\sim l$ occurs nowhere.
  Such a clause can never help to derive [ ].

- Tautologies

  Clauses which contain both a literal and its negation
  are useless in deriving [ ].

- Subsumed clauses

  A clause which contains a superset of the literals of another clause:

  - only the shorter clause is needed to derive [];
  - can be generalized to allow for substitutions.

Strategies 2

2. Ordering strategies

   There are many ways to vary the order of search; a simple and effective way is the unit clause preference

   > Always try to resolve with unit clauses first, since those always “shorten” the resolvent.

3. Set of support

   KB is usually satisfiable. Thus it makes little sense to resolve two clauses whose ancestors are from the KB only.

   Contradictions result from the interaction with the negation of the query ($\neg$F).

   Hence it is appropriate to require that one of the input clauses has an ancestor in $\neg$F.

Strategies 3

4. Operators with a sense of direction

   $[\sim p, q]$ can be interpreted in two ways.

   from p infer q (forward)
   to derive q first derive p (backward)

   In the first case

   one would only resolve $[\sim p, q]$ with $[p, \ldots]$ to obtain $[q, \ldots]$.

   In the second case

   one would only resolve $[\sim p, q]$ with $[\sim q, \ldots]$ to obtain $[\sim p, \ldots]$.