6.825 Techniques in Artificial Intelligence

## Resolution Theorem Proving: Propositional Logic

- Propositional resolution
- Propositional theorem proving
- Unification



## Propositional Resolution

- Resolution rule:
  - □ v □
  - ¬□ v □
  - ―――――
  - □ v □
- Resolution refutation:
  - Convert all sentences to CNF
  - Negate the desired conclusion (converted to CNF)
  - Apply resolution rule until either
    - – Derive false (a contradiction)
    - – Can't apply any more
- Resolution refutation is sound and complete
  - If we derive a contradiction, then the conclusion follows from the axioms
  - If we can't apply any more, then the conclusion cannot be proved from the axioms.



## Propositional Resolution Example

Prove R

| | |
|---|---|
| 1 | P v Q |
| 2 | P → R |
| 3 | Q → R |

| Step | Formula | Derivation |
|---|---|---|
| 1 | P v Q | Given |
| 2 | ¬ P v R | Given |
| 3 | ¬ Q v R | Given |
| 4 | ¬ R | Negated conclusion |
| ~~5~~ | ~~Q v R~~ | ~~1,2~~ |
| 6 | ¬ P | 2,4 |
| 7 | ¬ Q | 3,4 |
| 8 | R | 5,7 |
| 9 | • | 4,8 |



## Propositional Resolution Example

Prove R

| | |
|---|---|
| 1 | P v Q |
| 2 | P → R |
| 3 | Q → R |

false v R

¬ R v false

―――――

false v false

| Step | Formula | Derivation |
|---|---|---|
| 1 | P v Q | Given |
| 2 | ¬ P v R | Given |
| 3 | ¬ Q v R | Given |
| 4 | ¬ R | Negated conclusion |
| 5 | Q v R | 1,2 |
| 6 | ¬ P | 2,4 |
| 7 | ¬ Q | 3,4 |
| 8 | R | 5,7 |
| 9 | • | 4,8 |



## Propositional Resolution Example

Prove R

| | |
|---|---|
| 1 | P v Q |
| 2 | P → R |
| 3 | Q → R |

false v R

¬ R v false

―――――

false v false

| Step | Formula | Derivation |
|---|---|---|
| 1 | P v Q | Given |
| 2 | ¬ P v R | Given |
| 3 | ¬ Q v R | Given |
| 4 | ¬ R | Negated conclusion |
| 5 | Q v R | 1,2 |
| 6 | ¬ P | 2,4 |
| 7 | ¬ Q | 3,4 |
| 8 | R | 5,7 |
| 9 | • | 4,8 |



## The Power of False

Prove Z

| | |
|---|---|
| 1 | P |
| 2 | ¬ P |

| Step | Formula | Derivation |
|---|---|---|
| 1 | P | Given |
| 2 | ¬ P | Given |
| 3 | ¬ Z | Negated conclusion |
| 4 | • | 1,2 |

Note that (P  ¬ P) → Z is valid

Any conclusion follows from a contradiction – and so strict logic systems are very brittle.

## Example Problem

Convert to CNF

Prove R

| | |
|---|---|
| 1 | (P → Q) → Q |
| 2 | (P → P) → R |
| 3 | (R → S) → ¬(S → Q) |

- ¬(¬ P v Q) v Q
- (P ¬ Q) v Q
- (P v Q) (¬ Q v Q)
- (P v Q)



## Example Problem

Convert to CNF

Prove R

| | |
|---|---|
| 1 | (P → Q) → Q |
| 2 | (P → P) → R |
| 3 | (R → S) → ¬(S → Q) |

- ¬(¬ P v Q) v Q
- (P ¬ Q) v Q
- (P v Q) (¬ Q v Q)
- (P v Q)

- ¬(¬ P v P) v R
- (P ¬ P) v R
- (P v R) (¬ P v R)



## Example Problem

Convert to CNF

Prove R

| | |
|---|---|
| 1 | (P → Q) → Q |
| 2 | (P → P) → R |
| 3 | (R → S) → ¬(S → Q) |

- ¬(¬ P v Q) v Q
- (P ¬ Q) v Q
- (P v Q) (¬ Q v Q)
- (P v Q)

- ¬(¬ P v P) v R
- (P ¬ P) v R
- (P v R) (¬ P v R)

- ¬(¬ R v S) v ¬ (¬ S v Q)
- (R ¬ S) v (S ¬ Q)
- (R v S) (¬ S v S) ←(R v ¬ Q) ←←(¬ S v ¬ Q)
- (R v S) (R v ¬ Q) ←←(¬ S v ¬ Q)



## Resolution Proof Example

Prove R

| | |
|---|---|
| 1 | (P → Q) → Q |
| 2 | (P → P) → R |
| 3 | (R → S) → ¬←(S → Q) |

| | | |
|---|---|---|
| 1 | P v Q | |
| 2 | P v R | |
| 3 | ¬ P v R | |
| 4 | R v S | |
| 5 | R v ¬ Q | |
| 6 | ¬ S v ¬ Q | |
| 7 | ¬ R | Neg |
| | | |
| | | |
| | | |
| | | |
| | | |



## Resolution Proof Example

Prove R

| | |
|---|---|
| 1 | (P → Q) → Q |
| 2 | (P → P) → R |
| 3 | (R → S) → ¬←(S → Q) |

| | | |
|---|---|---|
| 1 | P v Q | |
| 2 | P v R | |
| 3 | ¬ P v R | |
| 4 | R v S | |
| 5 | R v ¬ Q | |
| 6 | ¬ S v ¬ Q | |
| 7 | ¬ R | Neg |
| 8 | S | 4,7 |
| 9 | ¬ Q | 6,8 |
| 10 | P | 1,9 |
| 11 | R | 3,10 |
| 12 | • | 7,11 |



## Proof Strategies

- Unit preference: prefer a resolution step involving an unit clause (clause with one literal).
  - Produces a shorter clause – which is good since we are trying to produce a zero-length clause, that is, a contradiction.
- Set of support: Choose a resolution involving the negated goal or any clause derived from the negated goal.
  - We're trying to produce a contradiction that follows from the negated goal, so these are "relevant" clauses.
  - If a contradiction exists, one can find one using the set-of-support strategy.

## Recitation Problems

Using resolution refutation, prove the last sentence in each group from the rest of the sentences in the group.

$P$ □ $Q$
□$P$ □ $R$
□$Q$ □ □$R$

($P$ □ $Q$) ($R$ □ $S$)
($P$ □ $S$) ($R$ □ $Q$)

□($P$ □□$Q$) □(□$S$□□$T$)
□($T$ $Q$)
$U$ □ (□$T$ □ (□$S$□$P$))
□$U$

Use resolution refutation to do problem 6.5 from R&N.



## First-Order Resolution

∀ x. P(x) → Q(x)
P(A)
Q(A)

Syllogism:
All men are mortal
Socrates is a man
Socrates is mortal

uppercase letters: constants
lowercase letters: variables

∀ x. ¬ P(x) v Q(x)
P(A)
Q(A)

Equivalent by definition of implication

¬ P(A) v Q(A)
P(A)
Q(A)

Substitute A for x, still true

then

Propositional resolution

The key is finding the correct substitutions for the variables.



## Substitutions

P(x, F(y), B) : an atomic sentence

| Substitution instances | Substitution $\{v_1/t_1,\ldots, v_n/t_n\}$ | Comment |
|---|---|---|
| P(z, F(w), B) | {x/z, y/w} | Alphabetic variant |
| P(x, F(A), B) | {y/A} | |
| P(G(z), F(A), B) | {x/G(z), y/A} | |
| P(C, F(A), B) | {x/C, y/A} | Ground instance |

Applying a substitution:

subst({A/y},P(x, F(y), B)) = P(x, F(A), B)

P(x, F(y), B) {A/y} = P(x,F(A),B)



## Unification

- Expressions $□_1$ and $□_2$ are unifiable iff there exists a substitution s such that $□_1$ s = $□_2$ s
- Let $□_1$ = x and $□_2$ = y, the following are unifiers

| s | $□_1$ s | $□_2$ s |
|---|---|---|
| {y/x} | x | x |
| {x/y} | y | y |
| {x/F(F(A)), y/F(F(A))} | F(F(A)) | F(F(A)) |
| {x/A, y/A} | A | A |



## Most General Unifier

g is a most general unifier of $□_1$ and $□_2$ iff for all unifiers s, there exists s′ such that $□_1$ s = ($□_1$ g) s′ and $□_2$ s = ($□_2$ g) s′

| $□_1$ | $□_2$ | MGU |
|---|---|---|
| P(x) | P(A) | {x/A} |
| P(F(x), y, G(x)) | P(F(x), x, G(x)) | {y/x} or {x/y} |
| P(F(x), y, G(y)) | P(F(x), z, G(x)) | {y/x, z/x} |
| P(x, B, B) | P(A, y, z) | {x/A, y/B, z/B} |
| P(G(F(v)), G(u)) | P(x, x) | {x/G(F(v)), u/F(v)} |
| P(x, F(x)) | P(x, x) | No MGU! |



## Unification Algorithm

```
unify(Expr x, Expr y, Subst s){
  if s = fail, return fail
  else if x = y, return s
  else if x is a variable, return unify-var(x, y, s)
  else if y is a variable, return unify-var(y, x, s)
  else if x is a predicate or function application,
        if y has the same operator,
                return unify(args(x), args(y), s)
        else    return fail
  else                    ; x and y have to be lists
        return unify(rest(x), rest(y),
                        unify(first(x), first(y), s))
}
```

## Unify-var subroutine

Substitute in for var and x as long as possible, then add new binding

```
unify-var(Variable var, Expr x, Subst s){
  if var is bound to val in s,
        return unify(val, x, s)
  else if x is bound to val in s,
        return unify-var(var, val, s)
  else if var occurs anywhere in x, return fail
  else   return add({var/x}, s)
}
```

Note: last line incorrect in book!



## Unification Problems

For each pair of sentences, give an MGU.

| | |
|---|---|
| Color(Tweety, Yellow) | Color(x,y) |
| Color(Tweety, Yellow) | Color(x,x) |
| Color(Hat(John), Blue) | Color(Hat(y), x) |
| R(F(x), B) | R(y,z) |
| R(F(y), x) | R(x, F(B)) |
| R(F(y), y, x) | R(x, F(A), F(v)) |
| Loves(x, y) | Loves(y, x) |
| F(G(w), H(w, J(x, y))) | F(G(v), H(u, v)) |
| F(G(w), H(w, J(x, u))) | F(G(v), H(u, v)) |
| F(x, F(u, x)) | F(F(y, A), F(z, F(B,z))) |



## Inference using Unification

$\forall$ x. ¬ P(x) v Q(x)

P(A)

---

Q(A)

For universally quantified variables, find MGU {x/A} and proceed as in propositional resolution.