Resolution in First Order Logic

\[ P(F(s), G(F(A))) \lor Q(s) \]
\[ \neg P(t, G(t)) \]
Resolution in First Order Logic

\[(p \lor \alpha_1 \lor \ldots \lor \alpha_k) \lor (\neg p' \lor \beta_1 \lor \ldots \lor \beta_n)\]

\[\neg P(t, G(t))\]

Subst(\(\varepsilon\), \((\alpha_1 \lor \ldots \lor \alpha_k \lor \beta_1 \lor \ldots \lor \beta_n)\))

where \(\varepsilon = \text{Unify}(p, p')\)

and standardize variables again

\[\varepsilon = \text{Unify}(P(F(s), G(F(A))), P(t, G(t)))\]
Unification

Initial call: Unify(x,y,[])

Unify(x,y,b):
  If x=y then Return(b)
  Else if Variable(x) then UnifyVar(x,y,b)  Think [x/y] with boundary cases
  Else if Variable(y) then UnifyVar(y,x,b)  Think [y/x] with boundary cases
  Else if [predicate(x) OR function(x)] AND [predicate(y) OR function(y)]
      AND [Head(x)=Head(y)] AND [#args(x) = #args(y)] then
      For i=1 to #args(x)
          b ← Unify(Arg(x,i),Arg(y,i),b)
          If b=“FAIL” then Return(“FAIL”)
      Return(b)
  Else Return(“FAIL”)

Unify(P(F(s),G(F(A))), P(t,G(t)),[])
Unification

Initial call: Unify(x,y,[])

Unify(x,y,b):
    If x=y then Return(b)
    Else if Variable(x) then UnifyVar(x,y,b)    Think [x/y] with boundary cases
    Else if Variable(y) then UnifyVar(y,x,b)    Think [y/x] with boundary cases
    Else if [predicate(x) OR function(x)] AND [predicate(y) OR function(y)]
           AND [Head(x)=Head(y)] AND [#args(x) = #args(y)] then
        For i=1 to #args(x)
             b ← Unify(Arg(x,i),Arg(y,i),b)
             If b=“FAIL” then Return(“FAIL”)
        Return(b)
    Else Return(“FAIL”)

Unify( P(F(s),G(F(A))), P(t,G(t)) ,[])
Unify( F(s), t, [])

Unify( F(s), t, [])
Unification

UnifyVar(var,u,b):

If “var/v” is in b for some v then Return(Unify(v,u,b))
Else if “u/v” is in b for some v then Return(Unify(var,v,b))
Else if var occurs inside u then Return(“FAIL”)
Else Return( b + [var/u] )
Unification

Initial call: Unify(x,y,[])

Unify(x,y,b):
  If x=y then Return(b)
  Else if Variable(x) then UnifyVar(x,y,b)  Think [x/y] with boundary cases
  Else if Variable(y) then UnifyVar(y,x,b)  Think [y/x] with boundary cases
  Else if [predicate(x) OR function(x)] AND [predicate(y) OR function(y)]
    AND [Head(x)=Head(y)] AND [#args(x) = #args(y)] then
    For i=1 to #args(x)
      b ← Unify(Arg(x,i),Arg(y,i),b)
      If b="FAIL" then Return("FAIL")
    Return(b)
  Else Return("FAIL")

Unify( P(F(s),G(F(A))), P(t,G(t)) ,[])

b = [t/F(s)]
Unification

Initial call: Unify(x,y,[])

Unify(x,y,b):
   If x=y then Return(b)
   Else if Variable(x) then UnifyVar(x,y,b)  Think [x/y] with boundary cases
   Else if Variable(y) then UnifyVar(y,x,b)  Think [y/x] with boundary cases
   Else if [predicate(x) OR function(x)] AND [predicate(y) OR function(y)]
   AND [Head(x)=Head(y)] AND [#args(x) = #args(y)] then
   For i=1 to #args(x)
      b ← Unify(Arg(x,i),Arg(y,i),b)  b = [t/F(s)]
      If b="FAIL" then Return("FAIL")
   Return(b)
   Else Return("FAIL")

Unify( P(F(s),G(F(A))), P(t,G(t)) ,[])
Unify( G(F(A)), G(t), [t/F(s)])
Unification

Unify(x, y, b):
If x = y then Return(b)
Else if Variable(x) then UnifyVar(x, y, b)  Think [x/y] with boundary cases
Else if Variable(y) then UnifyVar(y, x, b)  Think [y/x] with boundary cases
Else if [predicate(x) OR function(x)] AND [predicate(y) OR function(y)]
    AND [Head(x) = Head(y)] AND [\#args(x) = \#args(y)] then
    For i = 1 to \#args(x)
        b ← Unify(Arg(x, i), Arg(y, i), b)
    If b = “FAIL” then Return(“FAIL”)
    Return(b)
Else Return(“FAIL”)

Initial call: Unify(x, y, [])

Unify( P(F(s), G(F(A))), P(t, G(t)), [])
    Unify( G(F(A)), G(t), [t/F(s)])
    Unify( F(A), t, [t/F(s)])
Unification

UnifyVar(var, u, b):

If “var/v” is in b for some v then Return(Unify(v, u, b))
Else if “u/v” is in b for some v then Return(Unify(var, v, b))
Else if var occurs inside u then Return(“FAIL”)
Else Return( b + [var/u] )

UnifyVar(t, F(A), [t/F(s)])
Unification

UnifyVar(var,u,b):

If “var/v” is in b for some v then Return(Unify(v,u,b))
Else if “u/v” is in b for some v then Return(Unify(var,v,b))
Else if var occurs inside u then Return(“FAIL”)
Else Return( b + [var/u] )

UnifyVar(t, F(A), [t/F(s)])
Unify(F(A), F(s), [t/F(s)])
Unification

UnifyVar(var, u, b):

If “var/v” is in b for some v then Return(Unify(v, u, b))
Else if “u/v” is in b for some v then Return(Unify(var, v, b))
Else if var occurs inside u then Return(“FAIL”)
Else Return( b + [var/u] )
Unification

Initial call: Unify(x,y,[])

Unify(x,y,b):
  If x=y then Return(b)
  Else if Variable(x) then UnifyVar(x,y,b)  Think [x/y] with boundary cases
  Else if Variable(y) then UnifyVar(y,x,b)  Think [y/x] with boundary cases
  Else if [predicate(x) OR function(x)] AND [predicate(y) OR function(y)]
    AND [Head(x)=Head(y)] AND [#args(x) = #args(y)] then
      For i=1 to #args(x)
        b ← Unify(Arg(x,i),Arg(y,i),b)
        If b="FAIL" then Return("FAIL")
      b = [t/F(s), s/A]
      Return(b)
  Else Return("FAIL")
Resolution in First Order Logic

\[(\neg p \lor \beta_1 \lor \ldots \lor \beta_n)\]

Subst(\(\epsilon\), \((\alpha_1 \lor \ldots \lor \alpha_k \lor \beta_1 \lor \ldots \lor \beta_n)\))

where \(\epsilon = \text{Unify}(p, p')\)

and standardize variables again

\[\neg P(t, G(t))\]

\[\epsilon = \text{Unify}(P(F(s), G(F(A))), P(t, G(t)))\]

\[= [t/F(s), s/A]\]
Resolution in First Order Logic

\((p \lor \alpha_1 \lor ... \lor \alpha_k)\)

\((\neg p' \lor \beta_1 \lor ... \lor \beta_n)\)

Subst(\(\varnothing\), \((\alpha_1 \lor ... \lor \alpha_k \lor \beta_1 \lor ... \lor \beta_n)\))

where \(\varnothing = \text{Unify}(p, p')\)

and standardize variables again

\(P(F(s),G(F(A))) \lor Q(s)\)

\(\neg P(t,G(t))\)

Subst(\([t/F(s), s/A], Q(s)\)

\(\varnothing = \text{Unify(} P(F(s),G(F(A))), P(t,G(t)) \) \)

\(= [t/F(s), s/A] \)
Resolution in First Order Logic

\[(p \lor \alpha_1 \lor \ldots \lor \alpha_k)\]
\[\neg p' \lor \beta_1 \lor \ldots \lor \beta_n\]

\[
\begin{align*}
\text{Subst}(\varnothing, (\alpha_1 \lor \ldots \lor \alpha_k \lor \beta_1 \lor \ldots \lor \beta_n))
\text{ where } \varnothing &= \text{Unify}(p, p') \\
\text{and standardize variables again}
\end{align*}
\]

\[
\begin{align*}
P(F(s), G(F(A))) \lor Q(s) \\

\neg P(t, G(t))
\end{align*}
\]

\[
\begin{align*}
\text{Subst}( [t/F(s), s/A], Q(s) ) &= Q(A) \\
\varnothing &= \text{Unify} ( P(F(s), G(F(A))), P(t, G(t)) ) \\
&= [t/F(s), s/A]
\end{align*}
\]
Resolution in First Order Logic

\[(\neg p \lor \alpha_1 \lor \ldots \lor \alpha_k)\]

\[\neg \neg p \lor (\beta_1 \lor \ldots \lor \beta_n)\]

\[Subst(\theta, (\alpha_1 \lor \ldots \lor \alpha_k \lor \beta_1 \lor \ldots \lor \beta_n))\]

where \(\theta = \text{Unify}(p, p')\)

and standardize variables again

\[P(F(s), G(F(A))) \lor Q(s)\]

\[\neg P(t, G(t))\]

\[\text{Subst([t/F(s), s/A], Q(s)) = Q(A)}\]

\[\theta = \text{Unify}(P(F(s), G(F(A))), P(t, G(t)))\]

\[Q(A) = [t/F(s), s/A]\]
Resolution in First Order Logic

\[ P(F(s), G(F(A))) \lor Q(s) \]

\[ \neg P(t, G(t)) \]

\[ Q(A) \]
Resolution in First-Order Logic

• Conversion to CNF maintains satisfiability/unsatisfiability
  • Skolemization does something funny with semantics, maintains satisfiability
    [See book if you’re curious]

• Resolution is sound: If $\alpha \vdash \beta$ then $\alpha \models \beta$

• Resolution is refutation complete: If $\alpha \models \beta$ then $\text{CNF}(\alpha) \land \text{CNF}(\neg \beta) \vdash ()$
Resolution in First-Order Logic

• Conversion to CNF maintains satisfiability/unsatisfiability
  • Skolemization does something funny with semantics, maintains satisfiability
    [See book if you’re curious]

• Resolution is *sound*: If $\alpha \vdash \beta$ then $\alpha \models \beta$  [requires occur check]

• Resolution is *refutation complete*: If $\alpha \models \beta$ then $\text{CNF}(\alpha) \land \text{CNF}(\neg \beta) \vdash ()$
Resolution in First-Order Logic

• Conversion to CNF maintains satisfiability/unsatisfiability
  • Skolemization does something funny with semantics, maintains satisfiability
    [See book if you’re curious]

• Resolution is *sound*: If $\alpha \vdash \beta$ then $\alpha \models \beta$ [requires occur check]
• Resolution is *refutation complete*: If $\alpha \models \beta$ then $\text{CNF}(\alpha) \land \text{CNF}(\lnot \beta) \vdash ()$ [potentially exponential]
Resolution in First-Order Logic

• Conversion to CNF maintains satisfiability/unsatisfiability
  • Skolemization does something funny with semantics, maintains satisfiability
    [See book if you’re curious]

• Resolution is sound: If $\alpha \vdash \beta$ then $\alpha \models \beta$    [requires occur check]
• Resolution is refutation complete: If $\alpha \models \beta$ then $\text{CNF}(\alpha) \land \text{CNF}(\neg \beta) \vdash ()$
  [potentially exponential]

  What if $\alpha \not\models \beta$?
Resolution in First-Order Logic

• Conversion to CNF maintains satisfiability/unsatisfiability
  • Skolemization does something funny with semantics, maintains satisfiability
    [See book if you’re curious]

• Resolution is sound: If $\alpha \vdash \beta$ then $\alpha \models \beta$  
  [requires occur check]

• Resolution is refutation complete: If $\alpha \models \beta$ then $\text{CNF}(\alpha) \land \text{CNF}(\lnot \beta) \vdash ()$
  [potentially exponential]

What if $\alpha \not\models \beta$?

May never halt
Resolution in First-Order Logic

• Conversion to CNF maintains satisfiability/unsatisfiability
  • Skolemization does something funny with semantics, maintains satisfiability
    [See book if you’re curious]

• Resolution is *sound*: If $\alpha \vdash \beta$ then $\alpha \models \beta$ [requires occur check]

• Resolution is *refutation complete*: If $\alpha \models \beta$ then $\text{CNF}(\alpha) \land \text{CNF}(\neg \beta) \vdash ()$
  [potentially exponential]

  What if $\alpha \not\equiv \beta$?
  May never halt
  Resolution is *semi-decidable*
Resolution in First-Order Logic

• Conversion to CNF maintains satisfiability/unsatisfiability
  • Skolemization does something funny with semantics, maintains satisfiability
    [See book if you’re curious]

• Resolution is sound: If \( \alpha \vdash \beta \) then \( \alpha \models \beta \)  
  [requires occur check]

• Resolution is refutation complete: If \( \alpha \models \beta \) then \( \text{CNF}(\alpha) \land \text{CNF}(\neg \beta) \vdash () \)  
  [potentially exponential]

What if \( \alpha \not\models \beta \)?

May never halt

Resolution is semi-decidable
  [puts burden on user to be decidable and tractable]
Thus far: Don’t consider uncertainty
Thus far: Haven’t consider uncertainty

Next topic:
Expanding state-space search to reason about uncertainty
Thus far: Haven’t consider uncertainty

Next topic:
Expanding state-space search to reason about uncertainty
Markov Decision Processes
Chapters 17, 5 (Game Tree Search), 22
Markov Decision Process

Expands on State Space Search:

• States: \( S \) (including some Initial State)
• Actions: \( A \)

\[
P(s'|s, a) \quad s \in S, \ a \in A
\]

• Reward Function:

\[
R: S \times A \times S \rightarrow \mathbb{R}
\]
Markov Decision Process

Expands on State Space Search:

- States: \( S \) (including some Initial State)
- Actions: \( A \)

\[ P(s' | s, a) \ s \in S, \ a \in A \]

- Reward Function:

\[ R: S \times A \times S \rightarrow \mathbb{R} \] [Subsumes “deterministic” case]
Markov Decision Process

Expands on State Space Search:

• States: $S$ (including some Initial State)
• Actions: $A$

\[ P(s'|s, a): s \in S, a \in A \]  

• Reward Function:  

\[ R: S \times A \times S \rightarrow \mathbb{R} \]  

[Subsumes “deterministic” case]

• Goal State/Condition?
Markov Decision Process

Expands on State Space Search:

- States: $S$ (including some Initial State)
- Actions: $A$
  
  $$ P(s' \mid s, a) \quad s \in S, \ a \in A $$

- Reward Function:
  
  $$ R: S \times A \times S \rightarrow \mathbb{R} $$

- Goal State/Condition: Get lots of reward over time
Get lots of reward over time

Examples:

Reward = win/lose a game
Lots of reward = win lots of games

Reward = user clicks through on an ad/post
Lots of reward = get lots of click-throughs

Reward = user purchases a product
Lots of reward = get lots of purchases
Markov Decision Process

Expands on State Space Search:

• States: $S$ (including some Initial State)
• Actions: $A$
  \[ P(s' | s, a) \text{ where } s \in S, a \in A \]
• Reward Function:
  \[ R: S \times A \times S \rightarrow \mathbb{R} \]
• Goal State/Condition: Get lots of reward over time
Markov Decision Process

Vocabulary:

A *policy* ($\Pi$): tells you what to do in each state

\[ \Pi(s) \quad \Pi : S \rightarrow A \]
Markov Decision Process

Vocabulary:

A *policy* ($\Pi$): tells you what to do in each state

$$\Pi(s) \quad \Pi : S \rightarrow A$$

“Solving” a problem:
Instead of finding a solution path,
Find a policy $\Pi$ that gives <a lot of reward>
Example (Figure 17.1)

Initial State

Actions: Up, Down, Left Right
Example (Figure 17.1)

Initial State

Actions: Up, Down, Left Right

80% probability: go in direction you want
10% probability: 90° left of it
10% probability 90° right of it
Example (Figure 17.1)

Actions: Up, Down, Left Right

80% probability: go in direction you want
10% probability: 90° left of it
10% probability 90° right of it
If you hit a wall you stay where you are
Example (Figure 17.1)

\[ R(s, a, \langle 4, 3 \rangle) = 1 \]
\[ R(s, a, \langle 4, 2 \rangle) = -1 \]
Example (Figure 17.1)

\[
\begin{align*}
R(s,a,<4,3>) &= 1 \\
R(s,a,<4,2>) &= -1 \\
R(s,a,s') &= -0.04 \\
\text{for all other states}
\end{align*}
\]
Simplified Example

\[
R(s,a,<2,2>) = 1 \\
R(s,a,<2,1>) = -1 \\
R(s,a,s') = r = -.04 \\
\text{for all other states}
\]
Detailed Depiction of State Space

1,2
\[ R(s,a,<1,2>) = -0.04 \]

2,2
\[ R(s,a,<2,2>) = 1 \]

1,1
\[ R(s,a,<1,1>) = -0.04 \]

2,1
\[ R(s,a,<2,1>) = -1 \]
What Does “Get Lots of Reward” Mean?

One approach: Cumulative reward

Given: a sequence of states \([s_0, s_1, ..., s_t, ...]\) using actions \([a_0, a_1, ..., a_t, ...]\)

Cumulative reward:

\[
R(s_0, a_0, s_1) + R(s_1, a_1, s_2) + R(s_2, a_2, s_3) + ... = \sum_{t=0}^{\infty} R(s_t, a_t, s_{t+1})
\]
What Does “Get Lots of Reward” Mean?

One approach: Cumulative reward

Given: a sequence of states \([s_0, s_1, ..., s_t, ...]\) using actions \([a_0, a_1, ..., a_t, ...]\)

Cumulative reward:

\[
\sum_{t=0}^{\infty} R(s_t, a_t, s_{t+1})
\]
What Does “Get Lots of Reward” Mean?

One approach: Cumulative reward

Given: a policy \( \Pi \) (where \( a_t = \Pi(s_t), s_{t+1} = \text{apply}(\Pi(s_t), s_t) \))

Cumulative reward:

\[
\sum_{t=0}^{\infty} R(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t))
\]
What Does “Get Lots of Reward” Mean?

One approach: Cumulative reward

Given: a policy $\Pi$ (where $a_t = \Pi(s_t)$, $s_{t+1} = \text{apply}(\Pi(s_t), s_t)$)

Actions are probabilistic

Cumulative reward:

$$\sum_{t=0}^{\infty} R(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t)))$$
What Does “Get Lots of Reward” Mean?

One approach: Cumulative reward

Given: a policy \( \Pi \) (where \( a_t = \Pi(s_t), s_{t+1} = \text{apply}(\Pi(s_t), s_t) \))

Actions are probabilistic

Cumulative reward:

\[
E \left[ \sum_{t=0}^{\infty} R(s_t, \Pi(s_t), \text{apply}(\Pi(s_t), s_t)) \right]
\]