Resolution Proofs

- Requires axioms to be in clause form
- To do harder proofs, we convert their axioms to **clause form**
- **Clause form** is a disjunctions of literals
- How do we convert a set of axioms into clause form

Steps

1. Eliminate implications
2. Move negations down to the atomic formula
3. Eliminate existential quantifiers
4. Rename variables so that no two variables are the same
5. Move Universal quantifiers to the left
6. Move disjunctions down to the literals
7. Eliminate conjunctions
8. Rename all variables so that no two variables are the same
9. Eliminate Universal quantifiers

Example

- A(x)[Brick(x) → (E(y) [On (x, y) & !Pyramid(y)] & !E(y)[On(x, y) & On(y, x)] &A(y)[!Brick(y) → !Equal(x, y)])]

Step 1

- Eliminate Implication
- A(x)[!Brick(x) V (E(y)[On(x, y) &!Pyramid(y)] & !E(y)[On(x, y) & On(y, x)] & A(y)[Brick(y) V !Equal(x, y)])]

UNIVERSITY OF NEVADA RENO

Step 2

- Move negation down to the atomic formulas
- !A(x) [Exp(x)] → E(x)[!Exp(x)]
- !E(x) [Exp(x)] → A(x)[!Exp(x)]
- A(x)[!Brick(x) V
  (E(y)[On(x, y) &!Pyramid(y)] &
  A(y)[!(On(x, y) & On(y, x))] &
  A(y)[Brick(y) V !Equal(x, y)])]

Step 2 cont’d

- A(x)[!Brick(x) V
  (E(y)[On(x, y) &!Pyramid(y)] &
  A(y)[!On(x, y) V !On(y, x)] &
  A(y)[Brick(y) V !Equal(x, y)])]

Step 3

- Eliminate existential quantifiers
- E(x)[On(x, y) & !Pyramid(y)]

- On(x, Magic(x)) & !Pyramid(Magic(x))
- Magic is a **Skolem** function,
- On(x, Support(x)) & !Pyramid(Support(x))

UNIVERSITY OF NEVADA RENO

Step 4

- Rename variables – because next step is to move all universal quantifiers to the left
- A(x)[!Brick(x) V ((On(x, Support(x) &!Pyramid(Support(x))& A(y)[!On(x, y) V !On(y, x)] & A(z)[Brick(z) V !Equal(x, z)])]

Step 5

- Move universal quantifiers to the left
- A(x)A(y)A(z)[!Brick(x) V ((On(x, Support(x) &!Pyramid(Support(x))& !On(x, y) V !On(y, x) & Brick(z) V !Equal(x, z))]

Step 6

- Move disjunctions down to the literals
- A(x)A(y)A(z)[
  (!Brick(x) V (On(x, Support(x)) &
  !Pyramid(Support(x))) &
  (!Brick(x) V !On(x, y) V !On(y, x)) &
  (!Brick(x) V Brick(z) V !Equal(x, z))]

Step 6 cont'd

- A(x)A(y)A(z)[
(!Brick(x) V On(x, Support(x))) &
(!Brick(x) V !Pyramid(Support(x))) &
(!Brick(x) V !On(x, y) V !On(y, x)) &
(!Brick(x) V Brick(z) V !Equal(x, z))]

Step 7

- Eliminate Conjunctions by writing each part as a separate axiom
- A(x) [!Brick(x) V On(x, Support(x))]
- A(x) [!Brick(x) V !Pyramid(Support(x))]
- A(x)A(y)[!Brick(x) V !On(x, y) V !On(y, x)]
- A(x)A(z)[!Brick(x) V Brick(z)V !Equal(x,z)]

University of Nevada Reno

Step 8

- Rename variables
- A(x) [!Brick(x) V On(x, Support(x))]
- A(w) [!Brick(w) V !Pyramid(Support(w))]
- A(u)A(y)[!Brick(u) V !On(u, y) V !On(y, u)]
- A(v)A(z)[!Brick(v) V Brick(z)V !Equal(v,z)]

Step 9

- Eliminate universal quantifiers – assume all variables are universally quantified
- !Brick(x) V On(x, Support(x))
- !Brick(w) V !Pyramid(Support(w))
- !Brick(u) V !On(u, y) V !On(y, u)
- !Brick(v) V Brick(z) V !Equal(v, z)

Example

- On (B, A)
- On (A, Table)
- Let us show/prove that B is above the table
- Above (B, Table)
- We need a couple more relations

Example

- A(x)A(y)[On(x,y) ➔ Above(x, y)]
- A(x)A(y)A(z)[Above(x,y) & Above(y, z) ➔ Above(x, z)]

- !On(u, v) V Above(u, v)
- !Above(x,y) V !Above(y,z) V Above(x,z)

Example

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)

Resolve 2 and 5

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)

- Specialize
  - x to B
  - Z to Table
- !Above(B,y) V !Above(y,Table) V Above(B,Table)
- !Above(B, Table)

Result of Resolve(2,5)

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)
6. **!Above(B,y) V !Above(y,Table)**

Resolve 1 and 6

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)
6. !Above(B,y) V !Above(y,Table)

Specialize

- u to y (replace)
- v to Table

1. !On(y, Table) V Above(y, Table)
2. !Above(B,y) V !Above(y,Table)

Result of Resolve(1,6)

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)
6. !Above(B,y) V !Above(y,Table)
7. **!On(y, Table) V !Above(B,y)**

UNIVERSITY OF NEVADA RENO

Resolve 1 and 7

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)
6. !Above(B,y) V !Above(y,Table)
7. !On(y, Table) V !Above(B,y)

- Specialize
  - u to B
  - v to y (replace)

1. !On(B, y) V **Above(B, y)**
2. !On(y, Table) V **!Above(B,y)**

Result of Resolve(1,7)

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)
6. !Above(B,y) V !Above(y,Table)
7. !On(y, Table) V !Above(B,y)
8. **!On(B, y) V !On(y, Table)**

UNIVERSITY OF NEVADA RENO

Resolve 3 and 8

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)
6. !Above(B,y) V !Above(y,Table)
7. !On(y, Table) V !Above(B,y)
8. **!On(B, y) V !On(y, Table)**

- Specialize
  - y to A

1. On(B, A)
2. **!**On(B, A) V !On(A, Table)

Result of Resolve (3,8)

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)
6. !Above(B,y) V !Above(y,Table)
7. !On(y, Table) V !Above(B,y)
8. !On(B, y) V !On(y, Table)
9. **!On(A, Table)**

Resolve 4 and 9

1. !On(u, v) V Above(u, v)
2. !Above(x,y) V !Above(y,z) V Above(x,z)
3. On(B, A)
4. On(A, Table)
5. !Above(B, Table)
6. !Above(B,y) V !Above(y,Table)
7. !On(y, Table) V !Above(B,y)
8. !On(B, y) V !On(y, Table)
9. !On(A, Table)

- On (A, Table)
- !On(A, Table)
- Resolves to Nil
- You must be finished since you have arrived at a contradiction.
- Thus
  - !Above(B, Table) must be false
- Thus
  - **Above(B, Table) must be true**

UNIVERSITY OF NEVADA RENO

Issues

- Proof is exponential
- Resolution requires Unification (Consistent substitutions)
  - Rule: You can replace a variable by any term that does not contain the variable
  - Finding such substitutions is called UNIFICATION
- Theorem Provers make take too long
- Theorem Provers may not help you to solve practical problems even if they finish quickly
- Logic is weak as a representation for certain kinds of knowledge