Resolution Theorem Proving

- First-Order Logic Recap
- Conjunctive normal form
- The Resolution algorithm

Based on lecture notes from Dr. Matthew Hyde, 2010

First Order Logic

- Predicate symbols
- Man (John), Woman(Mary), Student (John)
- Mother (Mary, John)
- Brother (Pete, John)

First Order Logic

- Logical Connectives
- OR: V

  Man (John) V Woman(John)
- AND: Λ

  Brother (Pete, John) Λ Brother (John, Pete)
- NOT: ¬

  ¬Mother(Pete, John)
- IMPLIES: =>

  Mother(Mary, John) => Woman (Mary)

First Order Logic

- Exists ∃

  ∃x Mother(x, John)

  ∃y Bird(y) Λ ¬Flies(y)

- For all ∀

  ∀y King(y) => Man(y)

  ∀y Bird(y) => HasFeathers(y)

Inference in First Order Logic

- We can try to infer conclusions from the statements that we already know

  $\forall y$ King(y) $\wedge$ Greedy(y) => Evil(y)

  King(John)

  Greedy(John)

- Can we infer this?

  Evil(John)

Inference in First Order Logic

$\forall y$ King(y) $\wedge$ Greedy(y) => Evil(y)

King(John)

Greedy(John)

- We can infer "Evil(John)" if we use a unifier

{y/John}

- This puts 'John' where there is a variable 'y'
- The idea is to make two logical expressions look the same

Inference in First Order Logic

{y/John}

- The idea is to make two logical expressions look the same

  $\forall$y King(y) Λ Greedy(y) => Evil(y)

  $\forall$y King(John) Λ Greedy(John) => Evil(John)

  King(John)

  Greedy(John)

- We know: King(John) and Greedy(John) already
- **So we can infer Evil(John)**

Resolution in First Order Logic

- Resolution is one method for automated theorem proving
- It is important to AI because it helps logical agents to reason about the world
- It is one rule applied over and over

Resolution Algorithm

- Resolution proves new terms
  - Input a database and a statement
  - It negates the statement, adds that to the database, and then finds a contradiction if one exists
  - If it finds a contradiction, then the negated statement is false
  - Therefore, the original statement **must be true**

Resolution Algorithm

Key Idea

- Proof by Contradiction
- Proof by Refutation
- Reductio ad Absurdum
  - Literally: “reduction to an absurd thing”

Conjunctive Normal Form

- Resolution algorithm needs sentences in CNF

  $\forall y$ King(y) $\wedge$ Greedy(y) => Evil(y)

  ¬King(y) v ¬ Greedy(y) v Evil(y)

- Resolution applies to clauses
- Converting a knowledge base to CNF is easily automated

Resolution

- Applies one rule over and over **to clauses**
- Each pair that contains complementary clauses is resolved

- We have a knowledge base
- We have a question
- The resolution algorithm proves the question true or false

Resolution

- We want to prove that the set of clauses is unsatisfiable

A and ¬A is unsatisfiable

Asleep(you) Λ ¬Asleep(you)

FirstClass(exam) Λ ¬FirstClass(exam)

∀x Example

- Unification: replace the variables with the concrete instance
- ∀x asleep(x) => fail(x)
  - For all x, if x is asleep, x will fail
- asleep(you)
  - You are asleep
- fail(you)?
  - Will you fail?

∀x Example

- Convert first line to CNF

  ∀x asleep(x) => fail(x)

  ∀x ¬asleep(x) v fail(x)

  ¬asleep(x) v fail(x)

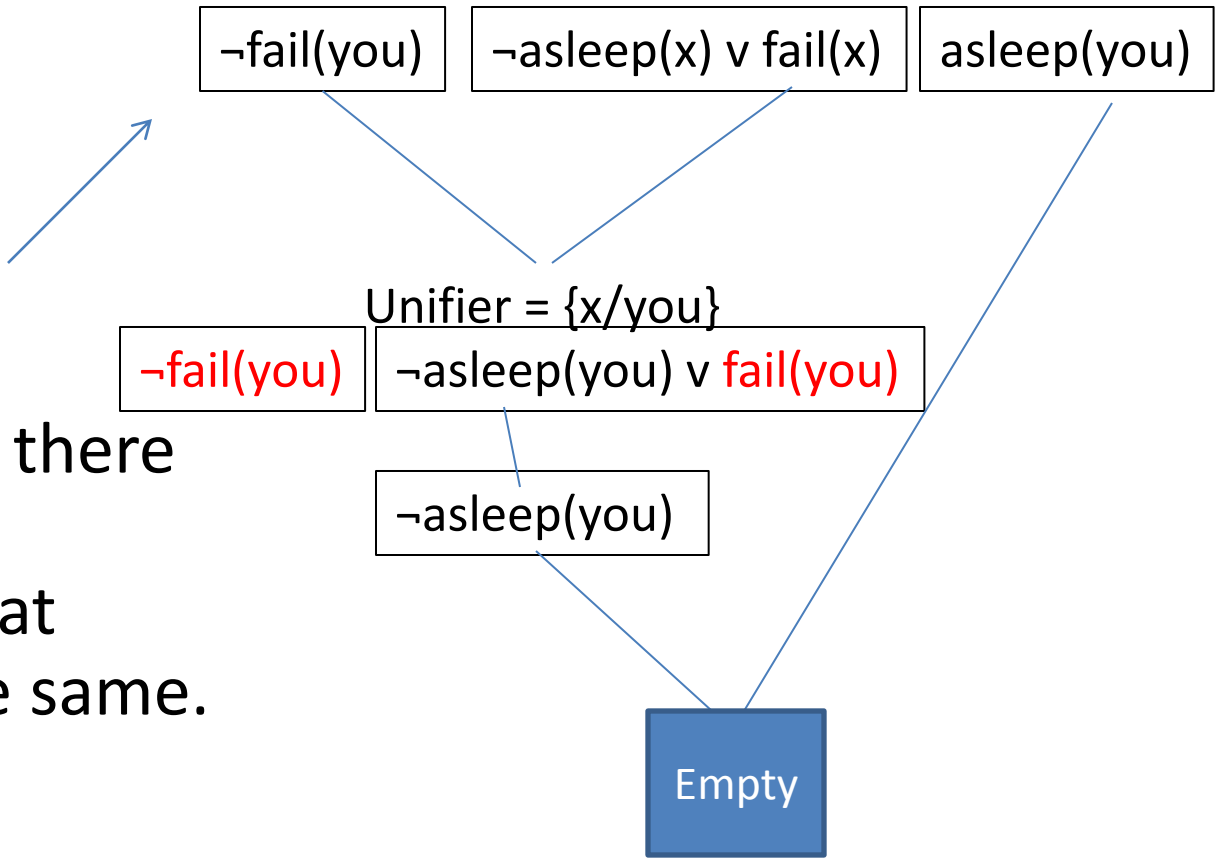
∀x Example

¬asleep(x) v fail(x)

asleep(you)

fail(you)?

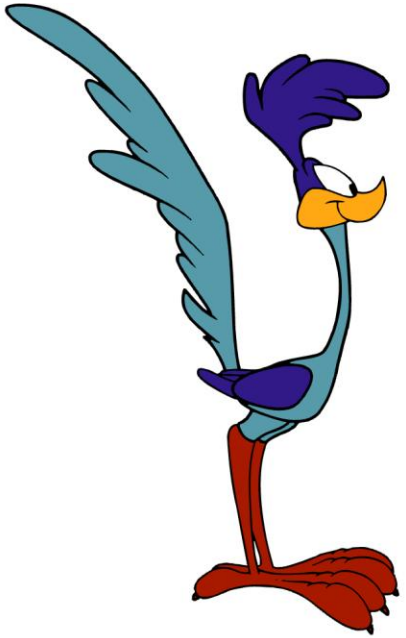
- Negate the goal
- Terms resolve if there is a set of substitutions that makes them the same. The unifier.

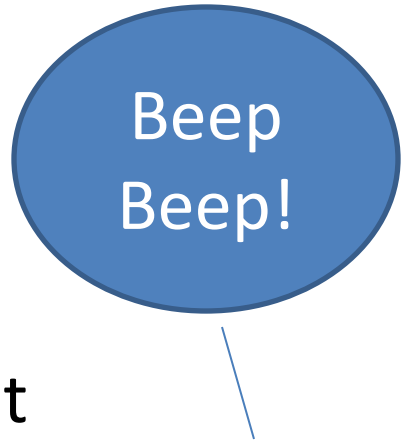


Skolemisation

- The process of removing existential quantifiers by elimination.
- $\exists x\ P(x)$ Skolemisation -> P(A), A: constant

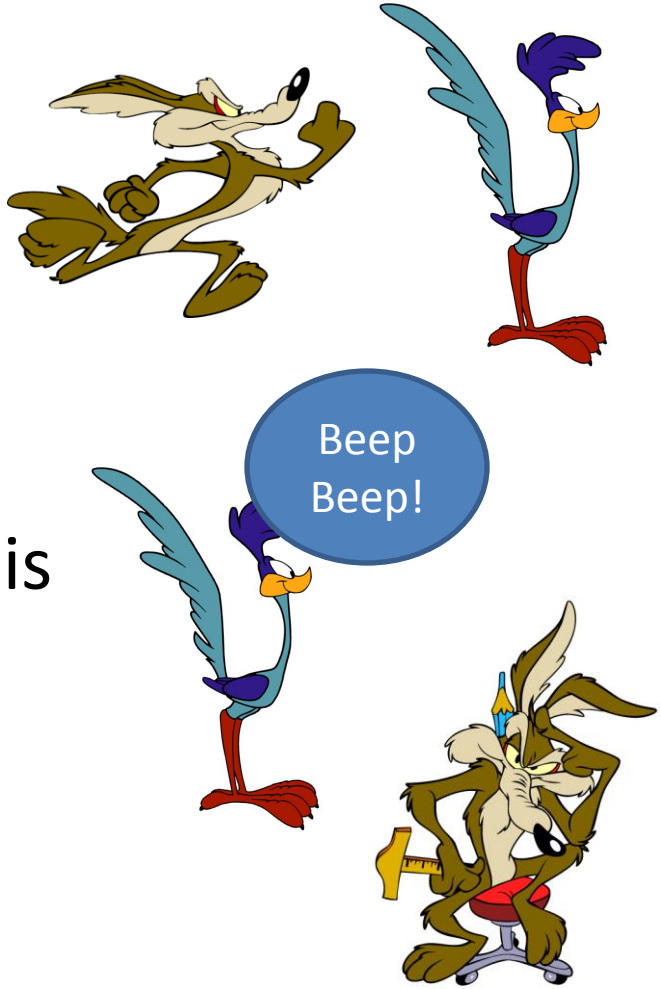
Roadrunner





Example - roadrunner

- Every coyote chases some roadrunner
- No coyote catches any smart roadrunner
- Any coyote who chases some roadrunner but does not catch it is frustrated
- All roadrunners are smart
- Question: Are all coyotes frustrated?



Example - roadrunner

Sentence	Knowledge Base
Every coyote chases some roadrunner	coyote(x) => rr(f(x))
	coyote(x) => chases(x,f(x))
No coyote catches any smart roadrunner	coyote(x) Λ rr(y) Λ smart(y) => ¬catches(x,y)
Any coyote who chases some roadrunner but does not catch it is frustrated	coyote(x) Λ rr(y) Λ chases(x,y) Λ ¬catches(x,y) => frustrated(x)
All roadrunners are smart	rr(x) => smart(x)
Question: Are all coyotes frustrated? (does there exist one coyote that isn't frustrated? If not then we have a proof by contradiction)	coyote(A)
	¬frustrated(A)

Example - roadrunner

Sentence	Knowledge Base
Every coyote chases some roadrunner	¬coyote(x) V rr(f(x))
	¬coyote(x) V chases(x,f(x))
No coyote catches any smart roadrunner	¬coyote(x) V ¬rr(y) V ¬smart(y) V ¬catches(x,y)
Any coyote who chases some roadrunner but does not catch it is frustrated	¬coyote(x) V ¬rr(y) V ¬chases(x,y) V catches(x,y) V frustrated(x)
All roadrunners are smart	¬rr(x) V smart(x)
Question: Are all coyotes frustrated? (does there exist one coyote that isn't frustrated? If not then we have a proof by contradiction)	coyote(A)
	¬frustrated(A)

Example - roadrunner

Knowledge Base
¬coyote(x) V rr(f(x))
¬coyote(x) V chases(x,f(x))
¬coyote(x) V ¬rr(y) V ¬smart(y) V ¬catches(x,y)
¬coyote(x) V ¬rr(y) V ¬chases(x,y) V catches(x,y) V frustrated(x)
¬rr(x) V smart(x)
coyote(A)
¬frustrated(A)

Example - roadrunner

Knowledge Base
¬coyote(x) V rr(f(x))
¬coyote(x) V chases(x,f(x))
¬coyote(x) V ¬rr(y) V ¬smart(y) V ¬catches(x,y)
¬coyote(x) V ¬rr(y) V ¬chases(x,y) V catches(x,y) V frustrated(x)
¬rr(x) V smart(x)
coyote(A)
¬frustrated(A)

Unifier = {x/A}

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬rr(y) V ¬smart(y) V ¬catches(A,y)
¬rr(y) V ¬chases(A,y) V catches(A,y) V frustrated(A)
¬rr(x) V smart(x)
coyote(A)
¬frustrated(A)

Unifier = {x/A}

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬rr(y) V ¬smart(y) V ¬catches(A,y)
¬rr(y) V ¬chases(A,y) V catches(A,y) V frustrated(A)
¬rr(x) V smart(x)
coyote(A)
¬frustrated(A)

Unifier = {y/f(A)}

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬rr(y) V ¬smart(y) V ¬catches(A,y)
¬chases(A,f(A)) V catches(A, f(A)) V frustrated(A)
¬rr(x) V smart(x)
coyote(A)
¬frustrated(A)

Unifier = {y/f(A)}

¬rr(f(A)) V ¬chases(A,f(A)) V catches(A,f(A)) V frustrated(A)

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬rr(y) V ¬smart(y) V ¬catches(A,y)
¬chases(A,f(A)) V catches(A, f(A)) V frustrated(A)
¬rr(x) V smart(x)
coyote(A)
¬frustrated(A)

Unifier = {y/f(A)}
¬rr(f(A)) V ¬smart(f(A)) V ¬catches(A,f(A))

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬smart(f(A)) V ¬catches(A,f(A))
¬chases(A,f(A)) V catches(A, f(A)) V frustrated(A)
¬rr(x) V smart(x)
coyote(A)
¬frustrated(A)

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬smart(f(A)) V ¬catches(A,f(A))
¬chases(A,f(A)) V catches(A, f(A)) V frustrated(A)
¬rr(x) V smart(x)
coyote(A)
¬frustrated(A)

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬smart(f(A)) V ¬catches(A,f(A))
¬chases(A,f(A)) V catches(A, f(A)) V frustrated(A)
smart(f(A))
coyote(A)
¬frustrated(A)

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬smart(f(A)) V ¬catches(A,f(A))
¬chases(A,f(A)) V catches(A, f(A)) V frustrated(A)
smart(f(A))
coyote(A)
¬frustrated(A)

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬catches(A,f(A))
¬chases(A,f(A)) V catches(A, f(A)) V frustrated(A)
smart(f(A))
coyote(A)
¬frustrated(A)

Example - roadrunner

Knowledge Base
rr(f(A))
chases(A,f(A))
¬catches(A,f(A))
¬chases(A,f(A)) V catches(A, f(A)) V frustrated(A)
smart(f(A))
coyote(A)
¬frustrated(A)

Example - roadrunner

Knowledge Base	
rr(f(A))	
chases(A,f(A))	
¬catches(A,f(A))	
frustrated(A)	new!
smart(f(A))	
coyote(A)	
¬frustrated(A)	

Contradiction!!

Example - roadrunner

frustrated(A) Λ ¬frustrated(A)

- This cannot be true, therefore our knowledge base cannot be true
- Question: If all roadrunners are smart, then all coyotes are frustrated?
- We added the opposite and proved it is not true. We proved that there is NOT at least ONE coyote that is NOT frustrated
- Therefore all coyotes are frustrated

Resolution Problems

- Can take a very long time
- Depending on the number of clauses in the knowledge base

- L1: King(y) V Greedy(y) V Evil(y) (covert first line to CNF)
- L2: King(John)
- L3: Greedy(John)
- L4: Evil(John) (negate the goal, add to knowledge base)
- L5:

What you need to know

- The steps to get a logic sentence into CNF
  - Including Skolemisation
- The resolution algorithm