Artificial Intelligence

First-order logic

Previously

- Propositional logic
  - Simplest language
  - Its world only consists of facts (and "explicit rules")
- Too puny a language to represent knowledge of complex environments with many objects in a concise way
  - Difficult to represent even the Wumpus world

$$B_{1,1} \Rightarrow P_{1,2} \vee P_{2,1}$$

Would like to say, "squares adjacent to pits are breezy" (not enumerate for all possible squares)

First-Order Logic

- Also called first-order predicate calculus
  - FOL, FOPC
- Makes stronger commitments
  - World consists of objects
    - Things with identities
    - e.g., people, houses, colors, …
  - Objects have properties/relations that distinguish them from other objects
    - e.g., Properties: red, round, square, …
    - e.g., Relations: brother of, bigger than, inside, …
  - Have functional relations
    - Return the object with a certain relation to given "input" object
    - The "inverse" of a (binary) relation
    - e.g., father of, best friend

Examples of Facts as Objects and Properties or Relations

- "Squares neighboring the Wumpus are smelly"
  - Objects
    - Wumpus, squares
  - Property
    - Smelly
  - Relation
    - Neighboring

Syntax of FOL: Basic Elements

- Constant symbols for specific objects

  KingJohn, 2, *OSU*, …
- Predicate (boolean) properties (unary) / relations (binary+)

  Brother(), Married(), >, …
- Functions (return objects)

  Sqrt() , *LeftLegOf()*, *FatherOf(), …*
- Variables

  x, y, a, b, …
- Connectives

  $\wedge$ $\vee$ $\neg$ $\Rightarrow$ $\Leftrightarrow$
- Equality

  $=$
- Quantifiers

  $\forall$ $\exists$

Atomic Sentences

- Collection of terms and relation(s) together to state facts
- Atomic sentence
  - *predicate*($term_1$, ..., $term_n$)
  - Or $term_1 = term_n$
- Examples

  Brother(Richard, John)

  Married(FatherOf(Richard), MotherOf(John))

Complex Sentences

- Made from atomic sentences using logical connectives

$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$

Examples:

Older(John, 30) $\Rightarrow \neg$*Younger(John, 30)*

$>(1,2) \ \vee \ \leq(1,2)$

Quantifiers

- Currently have logic that allows objects
- Now want to express properties of entire collections of objects
  - Rather than enumerate the objects by name
- Two standard quantifiers
  - Universal ∀
  - Existential ∃

Universal Qualification

- "For all …" (typically use implication $\Rightarrow$)
  - Allows for "rules" to be constructed
- $\forall$ <*variables*> <*sentence*>
  - Everyone at OSU is smart

    $\forall x \;\; At(x, OSU) \;\Rightarrow\; Smart(x)$
- $\forall x \; P$ is equivalent to <u>conjunction</u> of all <u>instantiations</u> of P

  $(At(John, OSU) \Rightarrow Smart(John))$

  $\wedge \;\; (At(Bob, OSU) \Rightarrow Smart(Bob))$

  $\wedge \;\; (At(Mary, OSU) \Rightarrow Smart(Mary)) \wedge \;\; \ldots$

Existential Quantification

- "There exists …" (typically use conjunction $\wedge$)
  - Makes a statement about some object (not all)
- $\exists$ <*variables*> <*sentences*>
  - Someone at OSU is smart

    $\exists x \;\; At(x, OSU) \wedge Smart(x)$
- $\exists\, x\, P$ is equivalent to disjunction of all instantiations of P

  $(At(John, OSU) \wedge Smart(John))$

  $\vee \;\; (At(Bob, OSU) \wedge Smart(Bob))$

  $\vee \; (At(Mary, OSU) \wedge Smart(Mary)) \;\vee\; \ldots$
- Uniqueness quantifier

  $\exists!\, x$ says a unique object exists (i.e. there is exactly one)

Properties of Quantifiers

- Quantifier duality: Each can be expressed using the other

$\forall x \;\; Person(x) \Rightarrow Likes(x, IceCream)$ "Everybody likes ice cream"

$\neg\exists x \;\; Person(x) \wedge \neg Likes(x, IceCream)$ "Not exist anyone who does not like ice cream"

$\exists x \; Person(x) \wedge Likes(x, Broccoli)$ "Someone likes broccoli"

$\neg\forall x \; Person(x) \Rightarrow \neg Likes(x, Broccoli)$ "Not the case that everyone does not like broccoli

Properties of Quantifiers

- Important relations

$\exists x \;\; P(x) \;\; = \;\; \neg \forall x \;\; \neg P(x)$

$\forall x \;\; P(x) \;\; = \;\; \neg \exists x \; \neg P(x)$

$P(x) \Rightarrow Q(x)$ is same as $\neg P(x) \vee Q(x)$

$\neg \; (P(x) \wedge Q(x))$ is same as $\neg P(x) \vee \neg Q(x)$

Proof

P	Q	$P \Rightarrow Q$
False	False	**TRUE**
False	True	**TRUE**
True	False	**FALSE**
True	True	**TRUE**

$$P(x) \Rightarrow Q(x)$$

is same as

$$\neg P(x) \vee Q(x)$$

P	Q	$\neg P$	$\neg P \vee Q$
False	False	True	**TRUE**
False	True	True	**TRUE**
True	False	False	**FALSE**
True	True	False	**TRUE**

Proof

P	Q	$P \wedge Q$	$\neg(P \wedge Q)$
False	False	False	**TRUE**
False	True	False	**TRUE**
True	False	False	**TRUE**
True	True	True	**FALSE**

$\neg(P(x) \wedge Q(x))$

is same as

$\neg P(x) \vee \neg Q(x)$

P	Q	$\neg P$	$\neg Q$	$\neg P \vee \neg Q$
False	False	True	True	**TRUE**
False	True	True	False	**TRUE**
True	False	False	True	**TRUE**
True	True	False	False	**FALSE**

Proof

P	Q	$P \vee Q$	$\neg(P \vee Q)$
False	False	False	**TRUE**
False	True	True	**FALSE**
True	False	True	**FALSE**
True	True	True	**FALSE**

$\neg(P(x) \vee Q(x))$

is same as

$\neg P(x) \wedge \neg Q(x)$

P	Q	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$
False	False	True	True	**TRUE**
False	True	True	False	**FALSE**
True	False	False	True	**FALSE**
True	True	False	False	**FALSE**

Conversion Example

1. $\forall x \;\; Person(x) \Rightarrow Likes(x, IceCream)$

[use: $\forall x \; P(x) = \neg\exists x \; \neg P(x)$]

2. $\neg\exists x \; \neg(Person(x) \Rightarrow Likes(x, IceCream))$

[use: $P(x) \Rightarrow Q(x)$ is same as $\neg P(x) \vee Q(x)$]

3. $\neg\exists x \; \neg(\neg Person(x) \vee Likes(x, IceCream))$

[distribute negatives]

4. $\neg\exists x \; Person(x) \wedge \neg Likes(x, IceCream)$

Nested Quantifiers

- $\forall x\ \forall y$ is same as $\forall y\ \forall x$ ($\forall x,y$)
- $\exists\ x\ \exists\ y$ is same as $\exists\ y\ \exists\ x$ ($\exists\ x,y$)
- $\exists\ x\ \forall y$ is not same as $\forall y\ \exists\ x$

$\exists\ y\ Person(y) \wedge (\forall x\ Person(x) \Rightarrow Loves(x,y))$

"There is someone who is loved by everyone"

$\forall x\ Person(x) \Rightarrow \exists y\ Person(y) \wedge Loves(x,y)$

"Everybody loves somebody"

(not guaranteed to be the same person)

Equality

- Equality symbol (=)
  - Make statements to the effect that two terms refer to the same object

"Henry is the Father of John"

$Father(John) = Henry$

"Spot has at least two sisters"

$\exists\, x,y\ Sister(x, Spot) \wedge Sister(y, Spot) \wedge \neg(x=y)$

More Sentences

- "Brothers are siblings"

  $\forall x,y \;\; Brother(x,y) \Rightarrow Sibling(x,y)$

- "One's mother is one's female parent"

  $\forall x,y \;\; Mother(x,y) \Rightarrow Female(x) \wedge Parent(x,y)$

Kinds of Rules

- For "Squares are breezy near a pit"
  - Diagnostic rule
    - Lead from observed effects to hidden causes
      - "Infer cause from effect"

$$\forall y \;\; Breezy(y) \Rightarrow \exists x \;\; Pit(x) \wedge Adjacent(x,y)$$

  - Causal "model-based" rule
    - Hidden world properties causes certain percepts
      - "Infer effect from cause"

$$\forall x,y \;\; Pit(x) \wedge Adjacent(x,y) \Rightarrow Breezy(y)$$

General Knowledge and Dealing with Categories

- Can we get further using less knowledge or more general knowledge?
- Classic example in AI is about knowledge and generality in sentences about birds and flight

Birds and Flight

- Organize facts about birds as listing of facts

(robins fly)	(gannets fly)	(western grebes fly)
(crows fly)	(penguins don't fly)	(ostriches don't fly)
(common loons fly)	(fulmars fly)	(arctic loons fly)

- Approximately 8,600 species of birds in world
  - Big list
  - Small in comparison to world population of ~100 billion birds!

Birds and Flight

- Rather than extending table, simpler to represent most facts with single symbol structure representing that birds of *all* species fly

$$\forall x, s \quad species(s, bird) \wedge inst(x, s) \Rightarrow flys(x)$$

- Reasoning about classes

Categorization

- Categorization is very basic cognitive mechanism
  - Treat different things as equivalent
  - Respond in terms of class membership rather than individuality
  - Fundamental to cognition and knowledge engineering
  - Reasoning by classes reduces complexity
- Overgeneralization
  - Treating all birds as equivalent about questions of flying
  - Need to handle exceptions
    - e.g., penguins (and ostriches) do not fly!

Category Exceptions

- How many circumstances determine whether individual birds can fly?
- Minsky, AAAI-85
  - "Cooked birds can't fly"
    - How about a cooked bird served in airline meal?
  - "Stuffed birds, frozen birds, and drowned birds cannot fly"
  - "Birds wearing concrete overcoats, and wooden bird decoys do not fly"
- Amount of special case knowledge increases
- Whether a particular bird can fly is determined by:
  - Its identity, condition, situation

Qualification Problem

- Want to separate typical statements from exceptions
- Qualification problem (McCarthy)
  - Proliferation of number of rules
  - Want to organize knowledge in general statements about usual cases
    - Categories are used to exploit regularities of the world
  - Then qualify statements describing their exceptions

Qualification Problem

- Abnormalcy predicates
  - Lay out qualifications for abnormal conditions
  - Example: "birds fly unless something abnormal about them"

    $(bird\ x)$ and $(not\ (ab_1\ x)) \rightarrow (flies\ x)$
  - Classes of birds that are abnormal and conditions

    $(disabled\text{-}bird\ x) \rightarrow (ab_1\ x)$
    $(fake\text{-}bird\ x) \rightarrow (ab_1\ x)$

    $(wears\ x\ concrete\text{-}overshoes) \rightarrow (disabled\text{-}bird\ x)$
    $(dead\ x) \rightarrow (disabled\text{-}bird\ x)$

    $(drowned\ x) \rightarrow (dead\ x)$
    $(stuffed\ x) \rightarrow (dead\ x)$
    $(cooked\ x) \rightarrow (dead\ x)$

    $(wooden\text{-}image\ x)$ and $(bird\ x) \rightarrow (fake\text{-}bird\ x)$

Categories and Exceptions

- Benefits of separating knowledge of typical from exceptions
  - Reduces number of sentences
  - Focus on categories of information
    - Guides introduction of new kinds of knowledge into categories to answer questions
- Basic goal to provide framework for assumptions
  - Default statements believed in absence of contradictory information
  - **"Unless you know otherwise for a particular bird, assume the bird can fly"**
- However, people have much richer model of what's going on for flying

Summary

- First-order logic
  - Increased expressive power over Propositional Logic
  - Objects and relations are semantic primitives
  - Syntax: constants, functions, predicates, equality, quantifiers
    - Two standard quantifiers
      - Universal ∀
      - Existential ∃
- Dealing with categories and exceptions
  - Qualification problem

Topics

- Reduction of first-order inference to propositional inference
- First-order inference algorithms
  - Generalized Modus Ponens
  - Forward chaining ***
  - Backward chaining ***
  - Resolution-based theorem proving ***

Reduction to Propositional Inference

Propositional vs. FOL Inference

- First-order inference can be done by converting KB to propositional logic and using propositional inference
  - Using modus ponens, etc.
- Specifically, what to do with quantifiers?
- Substitution: {*variable*/*Object*}
  - Remove quantifier by substituting *variable* with specific object

Reduction to Propositional Inference

- Universal Quantifiers ($\forall$)
  - Recall: Sentence must be true *for all* objects in the world (all values of variable)
  - So substituting any object must be valid (Universal Instantiation, UI)
- Example
  - 1) $\forall x \;\; Person(x) \rightarrow Likes(x, IceCream)$
  - Substituting: (1), $\{x/Jack\}$
  - 2) $Person(Jack) \rightarrow Likes(Jack, IceCream)$

Reduction to Propositional Inference (con't)

- Existential Quantifiers (∃)
  - Recall: Sentence must be true *for some* object in the world (or objects)
  - Assume we know this object and give it an arbitrary (unique!) name (Existential Instantiation, EI)
  - Known as Skolem constant (SK1, SK2, …)
- Example
  - 1) $\exists x \;\; Person(x) \wedge Likes(x,IceCream)$
  - Substituting: (1), $\{x/SK1\}$
  - 2) $Person(SK1) \wedge Likes(SK1,IceCream)$
- We don't know who "SK1" is (and usually can't), but we know they must exist

Reduction to Propositional Inference (con't)

- Multiple Quantifiers
  - No problem if same type ($\forall x,y$ or $\exists x,y$)
  - Also no problem if: $\exists x\ \forall y$
    - There must be some x for which the sentence is true with every possible y
    - Skolem constant still works (for x)
- Problem with $\forall x\ \exists y$
  - For every possible x, there must be some y that satisfies the sentence
  - Could be different y value to satisfy for each x!

Reduction to Propositional Inference (con't)

- Problem with $\forall x \exists y$ (con't)
  - The value we substitute for y must depend on x
  - Use a Skolem function instead
- Example
  - 1) $\forall x \exists y \; Person(x) \rightarrow Loves(x,y)$
  - Substitute: (1), $\{y/SK1(x)\}$
  - 2) $\forall x \; Person(x) \rightarrow Loves(x,SK1(x))$
  - Then: (2), $\{x/Jack\}$
  - 3) $Person(Jack) \rightarrow Loves(Jack,SK1(Jack))$
- $SK1(x)$ is *effectively* a function which returns a person that x loves. But, again, we can't generally know the specific value it returns.

Reduction to Propositional Inference (con't)

- Internal Quantifiers
  - Previous rules only work if quantifiers are external (left-most)
  - Consider: $\forall x\ (\exists y\ Loves(x,y)) \rightarrow Person(x)$
  - "For all x, if there is some y that x loves, then x must be a person"
  - A Skolem function limits the values y could take (to one) and we can't know what it is.
- Need to move the quantifier outward
  - $\forall x\ (\exists y\ Loves(x,y)) \rightarrow Person(x)$
  - $\forall x\ \neg(\exists y\ Loves(x,y)) \vee Person(x)$ (convert to $\neg,\vee,\wedge$)
  - $\forall x\ \forall y\ \neg Loves(x,y) \vee Person(x)$ (move $\neg$ inward)
  - $\forall x\ \forall y\ Loves(x,y) \rightarrow Person(x)$
- Now we can see that we can actually substitute *anything* for y
- May need to rename variables before moving quantifier left

Reduction to Propositional Inference (con't)

- Once have non-quantified sentences (from quantified sentences using UI, EI), possible to reduce first-order inference to propositional inference
- Suppose KB contains:

  $\forall x \;\; King(x) \wedge Greedy(x) \rightarrow Evil(x)$

  $King(John)$

  $Greedy(John)$

  $Brother(Richard, John)$

- Using UI with $\{x/John\}$ and $\{x/Richard\}$, we get

  $King(John) \wedge Greedy(John) \rightarrow Evil(John)$

  $King(Richard) \wedge Greedy(Richard) \rightarrow Evil(Richard)$

Reduction to Propositional Inference (con't)

- Now the KB is essentially propositional:

  $King(John) \wedge Greedy(John) \rightarrow Evil(John)$

  $King(Richard) \wedge Greedy(Richard) \rightarrow Evil(Richard)$

  $King(John)$

  $Greedy(John)$

  $Brother(Richard, John)$

- Then can use propositional inference algorithms to obtain conclusions
  - Modus Ponens yields $Evil(John)$

$$\frac{\alpha, \quad \alpha \rightarrow \beta}{\beta}$$

$$\frac{King(John) \wedge Greedy(John), \quad King(John) \wedge Greedy(John) \rightarrow Evil(John)}{Evil(John)}$$

Forward and Backward Chaining

Forward and Backward Chaining

- Have language representing knowledge (FOL) and inference rules (Generalized Modus Ponens)
  - Now study how a reasoning program is constructed
- Generalized Modus Ponens can be used in two ways:

  #1) Start with sentences in KB and generate new conclusions (forward chaining)
  - **"Used when a new fact is added to database and want to generate its consequences"**

  or

  #2) Start with something want to prove, find implication sentences that allow to conclude it, then attempt to establish their premises in turn (backward chaining)
  - **"Used when there is a goal to be proved"**

Forward Chaining

- Forward chaining normally triggered by addition of new fact to KB (using TELL)
- When new fact *p* added to KB:
  - For each rule such that *p* unifies with a premise
    - If the other premises are known, then add the conclusion to the KB and continue chaining
  - Premise: Left-hand side of implication
    - Or, each term of conjunction on left hand side
  - Conclusion: Right-hand side of implication
- Forward chaining uses unification
  - Make two sentences (fact + premise) match by substituting variables (if possible)
- Forward chaining is data-driven
  - Inferring properties and categories from percepts

Forward Chaining Example

- Add facts 1, 2, 3, 4, 5, 7 in turn
  - Number in [] is unification literal; √ rule firing

1. $Buffalo(x) \wedge Pig(y) \rightarrow Faster(x, y)$
2. $Pig(y) \wedge Slug(z) \rightarrow Faster(y, z)$
3. $Faster(x, y) \wedge Faster(y, z) \rightarrow Faster(x, z)$
4. $Buffalo(Bob)$ [1a, ×]
5. $Pig(Pat)$ [1b, √]
   6. $Faster(Bob, Pat)$ [3a, ×], [3b, ×]
      [2a, ×]
7. $Slug(Steve)$ [2b, √]
   8. $Faster(Pat, Steve)$ [3a, ×], [3b, √]
      9. $Faster(Bob, Steve)$ [3a, ×], [3b, ×]

Note: $\forall x,y,z$ dropped

Can't satisfy (1b), failed to fire rule

Also can satisfy (1a), can fire rule!

Another Example

Knowledge Base

$A \rightarrow B$

$A \rightarrow D$

$D \rightarrow C$

$A \rightarrow E$

$D \rightarrow F$

$E \rightarrow G$

Add A:

A, $A \rightarrow B$ gives B [done]

A, $A \rightarrow D$ gives D

D, $D \rightarrow C$ gives C [done]

D, $D \rightarrow F$ gives F [done]

A, $A \rightarrow E$ gives E

E, $E \rightarrow G$ gives G [done]

[done]

Order of generation B, D, C, F, E, G

Backward Chaining

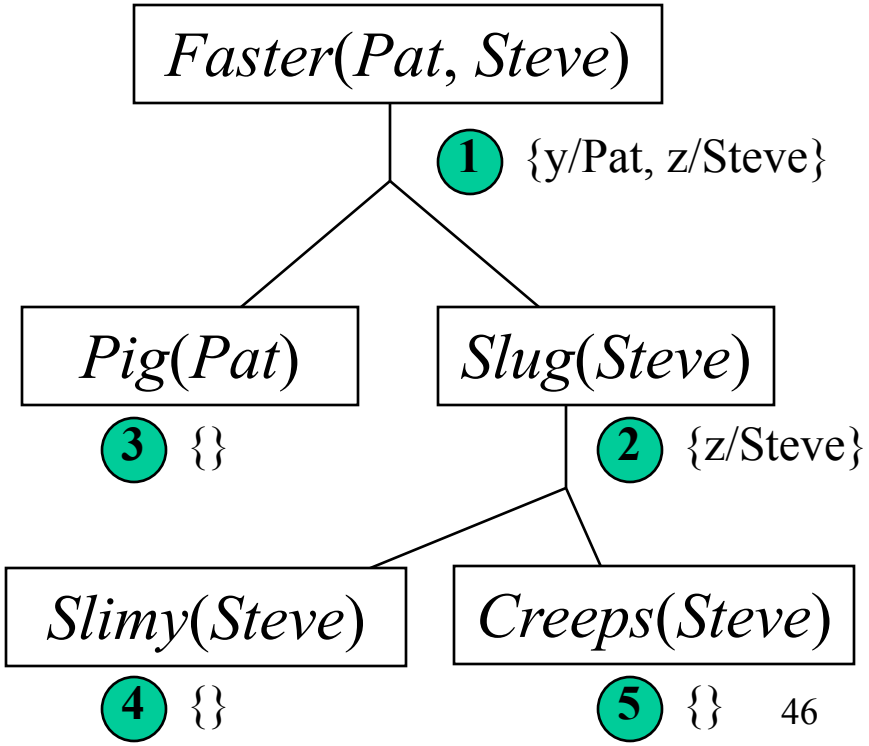
- Backward chaining designed to find all answers to a question posed to KB (using ASK)
- When a query *q* is asked:
  - If a matching fact *q'* is known, return the unifier
  - For each rule whose consequent *q'* matches *q*
    - Attempt to prove each premise of the rule by backward chaining
- Added complications
  - Keeping track of unifiers, avoiding infinite loops
- Two versions
  - Find any solution
  - Find all solutions
- Backward chaining is basis of logic programming
  - Prolog

Backward Chaining Example

Given facts/rules 1-5 in KB:

1. $Pig(y) \wedge Slug(z) \rightarrow Faster(y, z)$
2. $Slimy(z) \wedge Creeps(z) \rightarrow Slug(z)$
3. $Pig(Pat)$
4. $Slimy(Steve)$
5. $Creeps(Steve)$

Prove: $Faster(Pat, Steve)$



Resolution

Resolution

- Uses proof by contradiction
  - Referred to by other names
    - Refutation
    - Reductio ad absurdum
- Inference procedure using resolution
  - To prove *P*:
    - Assume *P* is FALSE
    - Add ¬*P* to KB
    - Prove a contradiction
  - Given that the KB is known to be True, we can believe that the negated goal is in fact False, meaning that the original goal must be True

Simple Example

- Given: "All birds fly", "Peter is a bird"
- Prove: "Peter flies"
- Step #1: have in FOL

$$\forall x \quad Bird(x) \rightarrow Flies(x)$$

$$Bird(Peter)$$

- Step #2: put in normal form

$$\neg Bird(x) \vee Flies(x)$$

$$Bird(Peter)$$

Simple Example (con't)

- Step #3: Assume contradiction of goal
  GOAL TO TEST: $\neg Flies(Peter)$
- Step #4: Unification $\{x/Peter\}$

$$\neg Bird(Peter) \vee Flies(Peter)$$

- Step #5: Resolution (unit)

$$\frac{\alpha, \neg\alpha \vee \beta}{\beta} \qquad \frac{\neg Flies(Peter), Flies(Peter) \vee \neg Bird(Peter)}{\neg Bird(Peter)}$$

- Step #6: Contradiction
  - The result of Step #5 says that "Peter is not a bird", but this is in contrast to KB containing *Bird*(*Peter*)
  - Therefore, we can conclude that "Peter does indeed fly"

KB:
$\neg Bird(x) \vee Flies(x)$
$Bird(Peter)$

Another Example

KB:

kb-1: $A(x,\text{bar}) \vee B(x) \vee C(x)$

kb-2: $D(y,\text{foo}) \vee \neg B(y)$

kb-3: $E(z) \vee \neg A(z,\text{bar})$

kb-4: $\neg D(\text{Minsky},\text{foo})$

kb-5: $\neg A(\text{Minsky},\text{bar})$

Goal: prove C(Minsky)

Another Example

KB:

kb-1: $A(x,\text{bar}) \vee B(x) \vee C(x)$

kb-2: $D(y,\text{foo}) \vee \neg B(y)$

kb-3: $E(z) \vee \neg A(z,\text{bar})$

kb-4: $\neg D(\text{Minsky},\text{foo})$

kb-5: $\neg A(\text{Minsky},\text{bar})$

Goal: prove C(Minsky)

0: $\neg C(\text{Minsky})$ *[add negated goal]*

1: $A(\text{Minsky},\text{bar}) \vee B(\text{Minsky}) \vee C(\text{Minsky})$ *[kb-1]*
{x/Minsky}

2: $\neg C(\text{Minsky}),\ A(\text{Minsky},\text{bar}) \vee B(\text{Minsky}) \vee C(\text{Minsky})$
2.a: $A(\text{Minsky},\text{bar}) \vee B(\text{Minsky})$ *[resolution: 0,1]*

3: $D(\text{Minsky},\text{foo}) \vee \neg B(\text{Minsky})$ *[kb-2]*
{y/Minsky}

4: $A(\text{Minsky},\text{bar}) \vee B(\text{Minsky}),\ D(\text{Minsky},\text{foo}) \vee \neg B(\text{Minsky})$
4.a: $A(\text{Minsky},\text{bar}) \vee D(\text{Minsky},\text{foo})$ *[resol: 2a,3]*

5: $\neg A(\text{Minsky},\text{bar}),\ A(\text{Minsky},\text{bar}) \vee D(\text{Minsky},\text{foo})$
5.a: $D(\text{Minsky},\text{foo})$ *[resol: 4a,kb-5]*

6: $D(\text{Minsky},\text{foo}) \wedge \neg D(\text{Minsky},\text{foo})$
FALSE, CONTRADICTION!!!
must be C(Minsky)

Summary

- Reduction of first-order inference to propositional inference
  - Universal and Existential Instantiation
- Forward chaining
  - Infer properties in data-driven manner
- Backward chaining
  - Proving query of a consequent by proving premises
- Resolution using proof by contradiction