Inference in FOPL

Artificial intelligence

Kristóf Karacs
PPKE-ITK

Logical inference schemes

- *Deduction*: formal logical reasoning
  - Premises: 1. All men are mortal. 2. Aristotle is a man.
  - Conclusion: Aristotle is mortal.
- *Induction*: generalization
  - Premise: The sun has risen in the east every morning up until now.
  - Conclusion: The sun will also rise in the east tomorrow.
- *Abduction*: choosing an explanation
  - Premise: 1. Flu causes fever. 2. Peter has fever.
  - Conclusion: Peter has flu.

The case of the silk gloves

"It was elementary my dear Watson. The killer always left a silk glove at the scene of the murder. That was his calling card. Our investigations showed that only three people have purchased such gloves in the past year. Of these, Professor Doolally and Reverend Fisheye have iron-clad alibis, so the murderer must have been Sergeant Heavyset. When he tried to murder us with that umbrella, we knew we had our man."

Not so elementary…

"The killer always left a silk glove at the scene of the murder.'" (induction)

"That was his calling card." (abduction)

"…only three people have purchased such gloves in the past year.'" (model generation)

"Professor Doolally and Reverend Fisheye have iron-clad alibis." (constraint based reasoning)

"…so the murderer must have been Sergeant Heavyset." (deduction)

First Order Predicate Logic (FOPL)

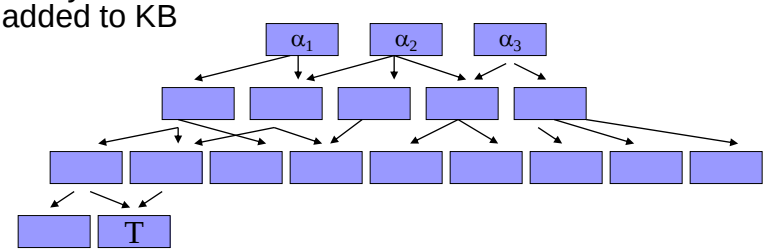
- Most used and analyzed logic
- Completeness: Gödel, Herbrand, 1930
  - If a FOPL statement is valid then it is provable
  - If $KB \models a$ then $KB \vdash a$
- Validity is semi-decidable
- Resolution: Robinson, 1963

Chains of inference

- Remember the problem we are trying to solve
  - Search for a path from axioms α_i, to theorem T
- Three approaches
  - Forward chaining
  - Backward chaining
  - Proof by contradiction
- Specification of a search problem:
  - Representation of states (first order predicate logic sentences)
  - Initial state (changes with the approach)
  - Operators (rules of inference, usually implication rules)
  - Goal state (changes with the approach)

Forward Chaining

- Start with initial axioms (atomic sentences) and deduce new facts by applying modus ponens
- Repeat until possible or query is answered
- Problems
  - Generates many irrelevant facts
  - Every rule has to be rechecked whenever a new fact is added to KB



Forward Chaining

- A first-order definite clause is a disjunction of literals of which exactly one is positive
  - Example
    - $white(X) \wedge potable(X) \rightarrow milk(X)$ is logically equivalent to $\neg white(X) \vee \neg potable(X) \vee milk(X)$
  - Modus ponens can be easily applied to first-order definite clauses
  - All variables are implicitly universally quantified
- Sound, complete

Backward Chaining

- Work backwards from the goal, chaining rules to find facts that support the conclusion
- For each node the inference rule has to be inverted
  - Which operator could have been applied to which state to produce this state (sentence)
- No problem when using equivalences
  - Can also use a bidirectional search (from both ends)
- Difficult when using implications
  - Many possible ways to invert operators

Proof By Contradiction

- "Reductio ad absurdum"
- Most often used method
- Idea: by showing that the assumption contradicts a set of axioms we can prove that the assumption is false

- KB' = Set of axioms (KB) + negated theorem (¬Th)
- If the ***F*** statement can be deduced from KB' then ¬Th is false, and thus Th must be true
- Advantage: heuristic function can be defined based on the distance from the 'False' statement

First order implication rules

- Propositional implications and equivalences
- First order implication rules
  - Quantifiers
  - Variables
  - Substitution

Universal elimination

- In a sentence φ any universally quantified variable *v* can be replaced by any ground term g

$$\frac{\forall v \ \varphi}{\text{subst}(\{v/g\}, \varphi)}$$

- Note: the variable has to be removed from quantification
- Example
  - $\forall x$ friend(Sue, *x*) becomes friend(Sue, Ann)

Existential introduction

- In a sentence φ any ground term g can be substituted by a variable *v* if it does not appear in φ

$$\frac{\varphi}{\exists v\ \mathrm{subst}(\{g/v\}, \varphi)}$$

- Example
  - friend(Sue, Ann) becomes $\exists x$ friend(Sue, *x*)
- Exercise
  - Find a sentence where *v* is in φ such that this implication rule is not sound

Universal introduction

- In a sentence φ any constant k can be substituted by a variable *v* if k is not mentioned in any of the premises or undischarged assumptions and *v* does not appear in φ

$$\frac{\varphi}{\forall v\ \mathrm{subst}(\{k/v\}, \varphi)}$$

- Example
  - friend(Sue, Doe) becomes $\forall x$ friend(Sue, *x*)

Existential elimination

- In a sentence φ any existentially quantified variable *v* can be replaced by any constant k, if k appears neither in φ nor anywhere else in the derivation

$$\frac{\exists v \ \varphi}{\text{subst}(\{v/k\}, \varphi)}$$

- k is called a Skolem constant
- Existential elimination is a special case of skolemization (see later)

Propositionalization

- Universal and existential elimination allow for inferring non-quantified sentences from quantified ones
- Reduces first-order inference to propositional inference
- Problem
  - Function symbols allow infinitely many ground terms: father(father (father (. . .)))
  - Can be overcome by Herbrand's theorem (R-N pp. 274–275)
- **Entailment** in FOPL is **semi-decidable** (Church)
  - Any entailed sentence can be proven
  - Not all false sentences can be disproven (Halting problem)

Inference with variables

- Premise
  - $\forall x$ (knows(Bob, x) $\rightarrow$ loves(Bob, x))
- From “knows(Bob, Alice)”
  - using modus ponens gives: “loves(Bob, Alice)”
- From “knows(Bryan, Alice)”
  - modus ponens cannot be used
- How to check applicability when variables are present?

Unifying predicates

- Expressions x_1 and x_2 are unifiable iff there exists a substitution Θ such that

  $$\text{subst}(\Theta, x_1) = \text{subst}(\Theta, x_2),$$

  where subst(Θ, x) applies Θ to x
- Unification by substitution ({X/Alice})
  - knows(Bob, X) and knows(Bob, Alice)
- Possibilities
  - variable-variable
  - variable-constant
  - variable-function

The unification algorithm

- A recursive algorithm
  - Passes around a set of substitutions, called *mu*
  - Makes sure that new substitutions are consistent with old ones
- `unify(x,y) = unify_internal(x,y,{})`
  - *x* and *y* can be variables, constants, lists, or compounds
- `unify_internal(x,y,mu)`
  - *x* and *y* are sentences, *mu* is a set of substitutions
  - finds substitutions making *x* look exactly like *y*
- `unify_variable(var,x,mu)`
  - *var* is a variable
  - finds a single substitution (which may be in *mu* already)

unify_internal

`unify_internal(x,y,mu)`

```
1.if (mu==failure) then return failure
2.if (x==y) then return mu
3.if (isa_variable(x)) then return
     unify_variable(x,y,mu)
4.if (isa_variable(y)) then return
     unify_variable(y,x,mu)
5.if (isa_compound(x) & isa_compound(y)) then return
     unify_internal(args(x),args(y),
            unify_internal(op(x),op(y),mu))
6.if (isa_list(x) & isa_list(y)) then return
     unify_internal(tail(x),tail(y),
            unify_internal(head(x),head(y),mu))
7.return failure
```

unify_variable

unify_variable(var,x,mu)

```
1. if (a substitution var/val is in mu) then
    return unify_internal(val,x,mu)
2. if (a substitution x/val is in mu) then
    return unify_internal(var,val,mu)
3. if (var occurs anywhere in x) return
    failure
4. add var/x to mu and return
```

Notes on the unification algorithm

- `unify_internal` will not match a constant to a constant, unless they are equal (case 2)
- Case 5 in `unify_internal` checks that two compound operators are the same (e.g., same predicate name)
- Case 6 in `unify_internal` causes the algorithm to recurse covering the whole list
- Cases 1 and 2 in `unify_variable` check that neither inputs have already been substituted

The occurs check

- When substituting variable x with an expression $f(x,y)$
  - x will be replaced by $f(x,y)$
  - But $f(x,y)$ still contains an instance of x, which has to be replaced again
  - We get $f(f(x,y),y)$, and then $f(f(f(x,y),y),y)$, etc.
  - Infinite recursion $\rightarrow$ the algorithm will not stop (halt)
- Case 3 in `unify_variable` checks this to avoid this situation
- Problem: Occurs check slows down the algorithm
  - Its complexity is $O(n^2)$, where n is the size of expressions being unified

Unification exercises

- nice(Alice) – nice(Mary)
- sees(x,Alice) – sees(y,Alice)
- sees(x,Alice) – sees(Mary,y)
- x – child(Alice,x)
- friends(x,y,Alice) $\wedge$ father(sonof(Bob),Bob) – father(z,Bob) $\wedge$ friends(Mary,z,u)
- R(F(y),x) – R(x,F(A))
- R(F(y),y,x) – R(x,F(A),F(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)))

Full resolution rule

- Resolution rules remove predicates in predicate logic
  - This is known as **resolving** the two sentences
- Unit resolution rule

$$\frac{(A \vee B),\ \neg B}{A}$$

- Full resolution rule (using CNF)

$$\frac{(A \vee B),\ (\neg B \vee C)}{A \vee C}$$

- With implication

$$\frac{(\neg A \rightarrow B),\ (B \rightarrow C)}{\neg A \rightarrow C}$$

Generalized resolution rule

- Given two CNF sentences

  $p_1 \vee p_2 \vee \ldots \vee p_m$ and $q_1 \vee q_2 \vee \ldots \vee q_n$

- If p_j and $\neg q_k$ can be unified, i.e. unify(p_j, $\neg q_k$) = Θ, then

$$\frac{p_1 \vee \ldots \vee p_j \vee \ldots p_m,\qquad q_1 \vee \ldots \vee q_k \vee \ldots \vee q_n}{\text{subst}(\Theta\ ,\ (p_1 \vee \ldots p_{j-1} \vee p_{j+1} \vee \ldots \vee p_m \vee q_1 \vee \ldots \vee q_{k-1} \vee q_{k+1} \vee \ldots q_n))}$$

Resolution with variables

- $P(x) \vee Q(x, y)$
- $\neg P(A) \vee R(B, z)$

- $\text{subst}(\{x/A\}, Q(x, y) \vee R(B, z))$
- $Q(A, y) \vee R(B, z)$

Local variable scope

- $P(x) \vee Q(x, y)$
- $\neg P(A) \vee R(B, x)$

Local variable scope

- $P(x_1) \vee Q(x_1, y)$
- $\neg P(A) \vee R(B, x_2)$

- $subst(\{x_1/A\}, Q(x_1, y) \vee R(B, x_2))$
- $Q(A, y) \vee R(B, x_2)$

CNF in FOPL

- Sentences need to be in conjunctive normal form (CNF)
  - Literals can contain variables, assumed to be universally quantified
- Example
  - $white(X) \wedge potable(X) \rightarrow milk(X)$ becomes $\neg white(X) \vee \neg potable(X) \vee milk(X)$

Conversion to clausal form

- 1. Eliminate $\rightarrow$ and $\leftrightarrow$
- 2. Drive in $\neg$ to atomic level
- 3. Rename variables apart
- 4. Skolemize
- 5. Drop universal quantifiers
- 6. Convert to CNF
- 7. Rename variables in each clause

Skolemization

- Substitute a new constant for each existentially quantified variable
  - $\frac{\exists x\ P(x)}{P(C_S)}$
- Substitute a new function of all universally quantified variables in enclosing scopes for each existentially quantified variable
  - $\frac{\forall x\ \exists y\ P(x, y)}{\forall x\ P(x, f_S(x))}$

“A cat called Tuna” (from textbook)

- Jack owns a dog
- Every dog owner is an animal lover
- No animal lover kills an animal.
- Either Jack or Curiosity killed the cat, who is named Tuna.
- Did Curiosity kill the cat?

A. $\exists x$ (Dog(x) $\wedge$ Owns(Jack,x))
B. $\forall x$ ((($\exists y$) Dog(y) $\wedge$ Owns(x, y)) $\rightarrow$ AnimalLover(x))
C. $\forall x$ (AnimalLover(x) $\rightarrow$ (($\forall y$) Animal(y) $\rightarrow$ ¬Kills(x,y)))
D. Kills(Jack,Tuna) $\vee$ Kills(Curiosity,Tuna)
E. Cat(Tuna)
F. $\forall x$ (Cat(x) $\rightarrow$ Animal(x))
G. Kills(Curiosity, Tuna)

Conversion to clausal form

- 1. Eliminate $\rightarrow$ and $\leftrightarrow$
- 2. Drive in ¬ to atomic level
- 3. Rename variables apart
- 4. Skolemize
- 5. Drop universal quantifiers
- 6. Convert to CNF
- 7. Rename variables in each clause

Sentence A & B

- **(A) $\exists x.$ Dog(x) $\wedge$ Owns(Jack,x)**
- Dog(D) $\wedge$ Owns(Jack,D)

- **(B) $\forall x.$ ($\exists y.$ Dog(y) $\wedge$ Owns(x,y)) $\rightarrow$ AnimalLover(x)**
- $\forall x.$ ($\neg\exists y.$ Dog(y) $\wedge$ Owns(x,y)) $\vee$ AnimalLover(x)
- $\forall x.$ $\forall y.$ $\neg$(Dog(y) $\wedge$ Owns(x,y)) $\vee$ AnimalLover(x)
- $\forall x.$ $\forall y.$ $\neg$Dog(y) $\vee$ $\neg$Owns(x,y) $\vee$ AnimalLover(x)
- $\neg$Dog(y) $\vee$ $\neg$Owns(x,y)) $\vee$ AnimalLover(x)

Sentence C & D

- **(C) $\forall x.$ AnimalLover(x) $\rightarrow$ ($\forall y.$ Animal(y) $\rightarrow$ $\neg$Kills(x,y))**
- $\forall x.$ $\neg$AnimalLover(x) $\vee$ ($\forall y.$ Animal(y) $\rightarrow$ $\neg$Kills(x,y))
- $\forall x.$ $\neg$AnimalLover(x) $\vee$ ($\forall y.$ $\neg$Animal(y) $\vee$ $\neg$Kills(x,y))
- $\neg$AnimalLover(x) $\vee$ $\neg$Animal(y) $\vee$ $\neg$Kills(x,y)

- **(D) Kills(Jack,Tuna) $\vee$ Kills(Curiosity,Tuna)**

Sentence E, F and neg. Th.

- **(E) Cat(Tuna)**
- **(F) $\forall x$. Cat(x) $\rightarrow$ Animal(x)**
- ¬Cat(x) $\vee$ Animal(x)
- **(Th) ¬Kills(Curiosity,Tuna)**

Solution

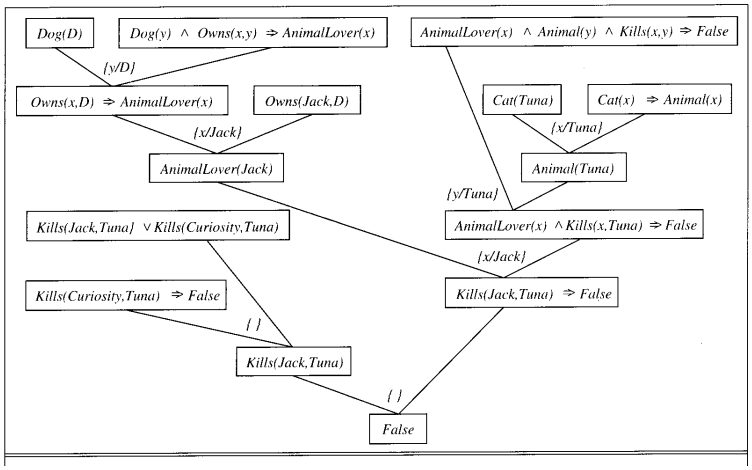
(D), (Th)	Kills(Jack,Tuna)	(G)
(E), (F), {x/T}	Animal(Tuna)	(H)
(C), (G), {x/J, y/T}	¬AnimalLover(Jack) $\vee$ ¬Animal(Tuna)	(I)
(H) , (I)	¬AnimalLover(Jack)	(J)
(B), (J), {x/J}	¬Dog(y) $\vee$ ¬Owns(Jack,y)	(K)
(A2)	¬Dog(D)	(L)
(A1), (L)	False	

CNF (Implicative form)

- Jack owns a dog
- Every dog owner is an animal lover
- No animal lover kills an animal.
- Either Jack or Curiosity killed the cat, who is named Tuna.
- Did Curiosity kill the cat?

A1. Dog(D)
A2. Owns(Jack,D)
B. Dog(y) $\wedge$ Owns(x,y) $\rightarrow$ AnimalLover(x)
C. AnimalLover(x) $\wedge$ Animal(y) $\rightarrow$ $\neg$Kills(x,y)
D. Kills(Jack,Tuna) $\vee$ Kills(Curiosity,Tuna)
E. Cat(Tuna)
F. $\neg$Cat(x) $\vee$ Animal(x)

Graph of proof



Equality

- Unification of different constants
  - Today(Thu), Today(Thursday)
- Expanding the KB is not sufficient
  - Thu = Thursday
- Extra axioms are needed
  - Equality is symmetric, reflexive and transitive
- Equality statements for each predicate:
  - $\forall x,y\ x = y \rightarrow (P(x) \leftrightarrow P(y))$ etc.

Demodulation rule

- Takes two input sentences, one expressing an equality ($\alpha = \beta$)
- Finds a unification for α with a term in another clause (Θ = unify(α, γ))
- Applies Θ to β (not α)
- Replaces occurrence of γ with Subst(Θ, β)

$$\frac{\alpha = \beta,\ (\ldots,\ \gamma,\ldots)}{(\ldots,\text{Subst}(\Theta,\ \beta),\ldots)}$$

Demodulation drawbacks

- Cannot bind variables in expression
  - father(Adam) = Bob
    - α: father(Adam) , β: Bob
  - older(father(x), x)
    - γ: father(x), Θ = {x/Adam}, Subst(Θ, β) = Bob
  - older(Bob, Adam) : not derived, only older(Bob, x)

$$\frac{\alpha = \beta, \quad (\ldots, \gamma, \ldots) \qquad \Theta = \text{unify}(\alpha, \gamma)}{(\ldots, \text{Subst}(\Theta, \beta), \ldots)}$$

- Equation must be a unit clause
  - $(x = \text{Adam} \wedge y = \text{Bob}) \rightarrow \text{father}(x) = y$
    - α cannot be father(x), since the equation is inside an implication
  - older(father(x), x)
  - $(x = \text{Adam} \wedge y = \text{Bob}) \rightarrow$ older(Bob, Adam)

Paramodulation

- F(x) = B
- Q(y) $\vee$ W(y,F(y))
- Q(y) $\vee$ W(y,B)

- $\alpha \vee (s = t)$
- $\beta \vee \gamma\,[r] \quad \Theta = \text{unify}(s,r)$
- Subst(Θ,($\alpha \vee \beta \vee \gamma\,[r]$))

- G(x) $\vee$ F(x) = B
- Q(y) $\vee$ W(y,F(y))
- G(y) $\vee$ Q(y) $\vee$ W(y,B)

- $s = F(x)$; $\quad t = B$
- $\gamma[\cdot] = W(y,\cdot)$; $\quad r = F(y)$
- Θ = {x/y}

Horn clauses

- Have the form: P1 $\wedge$ P2 $\wedge$ … $\wedge$ Pn $\rightarrow$ Q
- Special cases
  - P1 $\wedge$ P2 $\wedge$ … $\wedge$ P_n $\rightarrow$ False
  - True $\rightarrow$ Q
- Enables polynomial time inference
- Prolog (SLD resolution)
  - S: Selection function
  - L: Linear sequence of clauses
  - D: Definite clauses
  - Ordered resolution

Sample Prolog program

```
fun(X) :-
  red(X),
  car(X).

fun(X) :-
  blue(X),
  bike(X).
```

```
car(vw_beatle).
car(ford_escort).
bike(harley_davidson).
red(vw_beatle).
red(ford_escort).
blue(harley_davidson).

?-
  fun(harley_davidson).
yes
```

Resolution proving as search

- Search space: Sentences in FOPL
- Initial state: {KB, ¬Th.}
- Operator: generalized resolution inference rule
- Goal Check: Empty clause found
- Solution: two possibilities
  - Path from axioms to false clause (if we want proof)
  - Just the fact that we have reached the false clause (no proof required)

Elimination strategies

- Identical clause elimination
  - a resolution refutation without a clause occurring twice
- Pure literal elimination
  - A literal with no negated occurrence makes its clause superfluous
- Tautology elimination
  - No effect on satisfiability
- Subsumption elimination
  - Remove clauses that are more specific than others in the KB

Restriction strategies

- Unit resolution
  - One resolvent is always a unit clause (single literal)
- Input resolution
  - One resolved clause is always taken from initial KB
  - Complete, if the KB contains Horn clauses
- Linear resolution
  - One resolved clause is always taken from either the initial KB or from the ancestor of the other resolvent; Complete
- Set of Support
  - One resolvent is always taken from a subset of initial KB or from its descendant
  - Complete, if the clauses outside the SoS are satisfiable
- Ordered resolution
  - Clauses are treated as ordered sets, resolution is allowed only on the first literal

Applications of resolution

- Automated Theorem Proving (ATP)
- Proof verification
- Proof compression
- Automated Conjecture Making
- Interactive proving
- Proof planning

A famous example for ATP

- Axiomatization of Boolean algebra
- Standard axioms
  - $\forall a, b \in B\ a+b = b+a$
  - $\forall a, b, c \in B\ (a+b)+c = a+(b+c)$
  - $\exists 0 \in B$ (unit element for +) $0 + a = a$
  - $\forall a \in B\ \neg\neg a = a$
  - $\forall a \in B\ \neg(a + \neg a) = 0$
  - $\forall a, b, c \in B$
    $a + \neg(\neg b + \neg c) = \neg(\neg(a+b) + \neg(a+c))$

Robbins Problem

- Huntigton's proposal to axiomatize Boolean algebras (1933)
  - Commutativity + associativity
  - $\forall a, b \in B.\ a = \neg(\neg a + b) + \neg(\neg a + \neg b)$
- Herbert Robbins
  - Commutativity + associativity
  - $\forall a, b \in B.\ a = \neg(\neg(a + b) + \neg(a + \neg b))$
  - Got coined "Robbins algebra"

Solving the Robbins Problem

- William McCune and Larry Wos
  - Argonne National Laboratories
  - EQP & Otter (first order provers)
  - EQP solved this in 8 days, completed on Oct. 10,1996
  - One step from the proof:

    ¬(¬(¬(¬(¬(x) + x) + ¬(¬(x) + x) + x + x + x + x) +
    ¬(¬(¬(x) + x) + x + x + x) + x) =
    ¬(¬(¬(x) + x) + ¬(¬(x) + x) + x + x + x + x)
  - Otter proved that the proof is OK (its successor is called Prover9)

----- EQP 0.9, June 1996 -----
The job began on eyas09.mcs.anl.gov, Wed Oct 2 12:25:37 1996
UNIT CONFLICT from 17666 and 2 at 678232.20 seconds.
---------- PROOF ----------
2 (wt=7) [] -(n(x + y) = n(x)).
3 (wt=13) [] n(n(n(x) + y) + n(x + y)) = y.
5 (wt=18) [para(3,3)] n(n(n(x + y) + n(x) + y) + y) = n(x + y).
6 (wt=19) [para(3,3)] n(n(n(n(x) + y) + x + y) + y) = n(n(x) + y).
24 (wt=21) [para(6,3)] n(n(n(n(x) + y) + x + y + y) + n(n(x) + y)) = y.
47 (wt=29) [para(24,3)] n(n(n(n(n(x) + y) + x + y + y) + n(n(x) + y) + z) + n(y + z)) = z.
48 (wt=27) [para(24,3)] n(n(n(n(x) + y) + n(n(x) + y) + x + y + y) + y) = n(n(x) + y).
146 (wt=29) [para(48,3)] n(n(n(n(x) + y) + n(n(x) + y) + x + y + y + y) + n(n(x) + y)) = y.
250 (wt=34) [para(47,3)] n(n(n(n(n(x) + y) + x + y + y) + n(n(x) + y) + n(y + z) + z) + z) = n(y + z).
996 (wt=42) [para(250,3)] n(n(n(n(n(n(x) + y) + x + y + y) + n(n(x) + y) + n(y + z) + z) + z + u) + n(n(y + z) + u)) = u.
16379 (wt=21) [para(5,996),demod([3])] n(n(n(n(x) + x) + x + x + x) + x) = n(n(x) + x).
16387 (wt=29) [para(16379,3)] n(n(n(n(n(x) + x) + x + x + x) + x + y) + n(n(n(x) + x) + y)) = y.
16388 (wt=23) [para(16379,3)] n(n(n(n(x) + x) + x + x + x + x) + n(n(x) + x)) = x.
16393 (wt=29) [para(16388,3)] n(n(n(n(x) + x) + n(n(x) + x) + x + x + x + x) + x) = n(n(x) + x).
16426 (wt=37) [para(16393,3)] n(n(n(n(n(x) + x) + n(n(x) + x) + x + x + x + x) + x + y) + n(n(n(x) + x) + y)) = y.
17547 (wt=60) [para(146,16387)] n(n(n(n(n(x) + x) + n(n(x) + x) + x + x + x + x) + n(n(n(x) + x) + x + x + x) + x) + x) = n(n(n(x) + x) + n(n(x) + x) + x + x + x + x).
17666 (wt=33) [para(24,16426),demod([17547])] n(n(n(x) + x) + n(n(x) + x) + x + x + x + x) = n(n(n(x) + x) + x + x + x).
---------- end of proof ----------

A problem by Lewis Carol

- The only animals in this house are cats
- Every animal that loves to gaze at the moon is suitable for a pet
- When I detest an animal, I avoid it
- No animals are carnivorous unless they prowl at night
- No cat fails to kill a mice
- No animals ever like me, except those that are in this house
- Kangaroos are not suitable for pets
- None but carnivorous animals kill mice
- I detest animals that do not like me
- Animals that prowl at night always love to gaze at the moon
- Therefore, I always avoid a kangaroo

Summary

- FOPL semantics
- Chains of inference
- Propositionalization
- Resolution
  - Unification algorithm
  - Generalized resolution
  - Equality
  - Resolution strategies
- Automatic theorem proving