12

Inference in First-Order Logic

Claudia Chirita

School of Informatics, University of Edinburgh

THE UNIVERSITY *of* EDINBURGH informatics

Based on slides by: Jacques Fleuriot, Michael Rovatsos, Michael Herrmann, Vaishak Belle

12.a

Generalized Modus Ponens

GENERALIZED MODUS PONENS (GMP)

For the atomic sentences $p_1, \ldots, p_n, p'_1, \ldots, p'_n, q$, and a unifier θ s.t. $p'_i\theta = p_i\theta$ for all i, we have the inference rule:

$$\frac{p'_1, p'_2, \ldots, p'_n \qquad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \to q)}{q\theta}$$

GMP is used with KB of **definite clauses** (one positive literal).

All variables are assumed universally quantified.

EXAMPLE

p'_1 is $\mathrm{King}(\mathrm{John})$ p'_2 is $\mathrm{Greedy}(y)$

p_1 is $\mathrm{King}(x)$ p_2 is $\mathrm{Greedy}(x)$ q is $\mathrm{Evil}(x)$

θ is $(x/\mathrm{John}, y/\mathrm{John})$

$q\theta$ is $\mathrm{Evil}(\mathrm{John})$

We need to show that $p'_1, \ldots, p'_n, (p_1 \wedge \ldots \wedge p_n \rightarrow q) \models q\theta$, provided that $p'_i\theta = p_i\theta$, for all i and θ a unifier.

PROOF.

For any sentence p, we have that $p \models p\theta$ by the Universal Instantiation rule. Using this, we have:

1. $(p_1 \wedge \ldots \wedge p_n \rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \rightarrow q)\theta = (p_1\theta \wedge \ldots \wedge p_n\theta \rightarrow q\theta)$
2. $$\begin{aligned} p'_1, \ldots, p'_n \models p'_1 \wedge \ldots \wedge p'_n \models (p'_1 \wedge \ldots \wedge p'_n)\theta &= p'_1\theta \wedge \ldots \wedge p'_n\theta \\ &= p_1\theta \wedge \ldots \wedge p_n\theta \end{aligned}$$
   because by the definition of generalized modus ponens we have that $p'_i\theta = p_i\theta$, for all i.
3. From the previous two steps, and by applying modus ponens, $q\theta$ follows.

EXAMPLE · WINNIE-THE-POOH

It is known in The Hundred-Acre Wood that if someone who is very fond of food gives a treat to one of their friends, they must be really generous.

Eeyore, the sad donkey, has some hunny that he has received for his birthday from Winnie-the-Pooh, who, as we know, is very fond of food.

Prove that Winnie-the-Pooh is generous.

EXAMPLE · WINNIE-THE-POOH

It is an act of generosity for someone very fond of food to share treats with his friends.

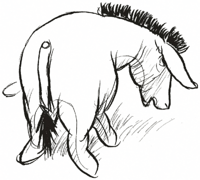
$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

Eeyore has some hunny.

$\exists x.\text{Owns}(\text{Eeyore}, x) \land \text{Hunny}(x)$

He must have received the hunny from Winnie-the-Pooh.

$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$



EXAMPLE · WINNIE-THE-POOH

Hunny is a treat.

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

Residents of The Hundred-Acre Wood are friends.

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

Eeyore is a resident of The Hundred-Acre Wood.

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

Pooh is very fond of food.

$\text{VeryFondOfFood}(\text{Pooh})$

EXAMPLE · WINNIE-THE-POOH

$\mathsf{VeryFondOfFood}(x) \land \mathsf{Treat}(y) \land \mathsf{Friend}(z) \land \mathsf{Gives}(x, y, z) \to \mathsf{Generous}(x)$

$\exists x.\mathsf{Owns}(\mathsf{Eeyore}, x) \land \mathsf{Hunny}(x)$ $\qquad$ $\mathsf{Owns}(\mathsf{Eeyore}, \mathsf{J}) \land \mathsf{Hunny}(\mathsf{J})$

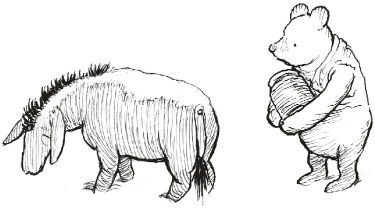
$\mathsf{Hunny}(x) \land \mathsf{Owns}(\mathsf{Eeyore}, x) \to \mathsf{Gives}(\mathsf{Pooh}, x, \mathsf{Eeyore})$

$\mathsf{Hunny}(x) \to \mathsf{Treat}(x)$

$\mathsf{Resident}(x, \mathsf{HundredAcreWood}) \to \mathsf{Friend}(x)$

$\mathsf{Resident}(\mathsf{Eeyore}, \mathsf{HundredAcreWood})$

$\mathsf{VeryFondOfFood}(\mathsf{Pooh})$



12.b

Forward chaining

function FOL-FC-ASK(KB, α) **returns** a substitution or *false*
  inputs: KB, the knowledge base, a set of first-order definite clauses
    α, the query, an atomic sentence
  local variables: new, the new sentences inferred on each iteration

  repeat until new is empty
    $new \leftarrow \{\}$
    for each $rule$ in KB **do**
      $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \leftarrow$ STANDARDIZE-VARIABLES($rule$)
      for each θ such that SUBST($\theta, p_1 \wedge \ldots \wedge p_n$) = SUBST($\theta, p'_1 \wedge \ldots \wedge p'_n$)
          for some $p'_1, \ldots, p'_n$ in KB
        $q' \leftarrow$ SUBST(θ, q)
        if q' does not unify with some sentence already in KB or new **then**
          add q' to new
          $\phi \leftarrow$ UNIFY(q', α)
          if ϕ is not *fail* **then return** ϕ
    add new to KB
  return *false*

Replaces all variables in its arguments with new ones

Pattern-matching

Facts irrelevant to the goal can be generated

FORWARD CHAINING · EXAMPLE

VeryFondOfFood$(x) \wedge$ Treat$(y) \wedge$ Friend$(z) \wedge$ Gives$(x, y, z) \rightarrow$ Generous(x)

Owns(Eeyore, J) $\wedge$ Hunny(J)

Hunny$(x) \wedge$ Owns(Eeyore, x) $\rightarrow$ Gives(Pooh, x, Eeyore)

Hunny$(x) \rightarrow$ Treat(x)

Resident(x, HundredAcreWood) $\rightarrow$ Friend(x)

Resident(Eeyore, HundredAcreWood)

VeryFondOfFood(Pooh)

VeryFondOfFood(Pooh) | Hunny(J) | Owns(Eeyore,J) | Resident(Eeyore,HAW)

FORWARD CHAINING · EXAMPLE

$\text{VeryFondOfFood}(x) \wedge \text{Treat}(y) \wedge \text{Friend}(z) \wedge \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J}) \wedge \text{Hunny}(\text{J})$

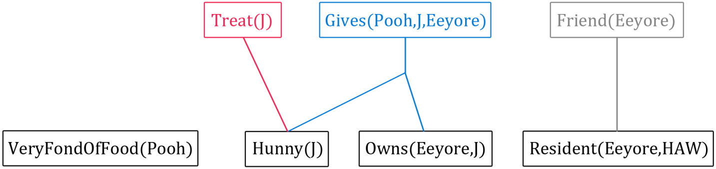
$\text{Hunny}(x) \wedge \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



FORWARD CHAINING · EXAMPLE

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J}) \land \text{Hunny}(\text{J})$

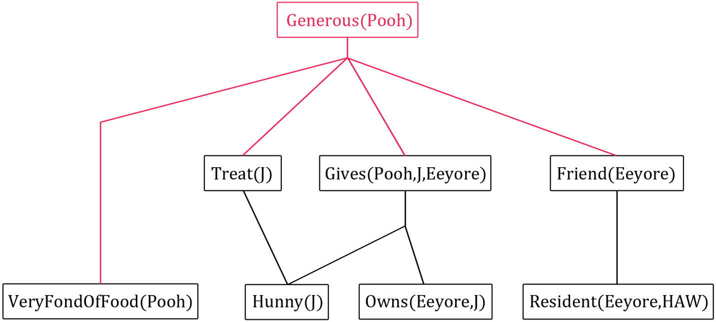
$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



Sound and complete for first-order **definite clauses** (clauses with exactly one positive literal).

Datalog $=$ first-order definite clauses $+$ no functions.

FC terminates for Datalog in a finite number of iterations.

May not terminate in general if the query q is **not** entailed.

Entailment with definite clauses is **semi-decidable**.

FORWARD CHAINING · EFFICIENCY

Incremental forward chaining

no need to match a rule on iteration k if a premise wasn't added on iteration $k-1$ $\Rightarrow$ match each rule whose premise contains a newly added positive literal.

Matching itself can be expensive:
Database indexing allows $O(1)$ retrieval of known facts.
 e.g. query Hunny(x) retrieves Hunny(J)

Forward chaining is widely used in **deductive databases**.

PATTERN MATCHING

For each θ s.t. SUBST$(\theta, p_1 \wedge \ldots \wedge p_n) =$ SUBST$(\theta, p_1' \wedge \ldots \wedge p_n')$ for some $p_1', \ldots, p_n'$ in KB

Finding all possible unifiers can be very expensive.

EFFICIENCY OF FORWARD CHAINING

EXAMPLE

$\text{Hunny}(x) \wedge \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

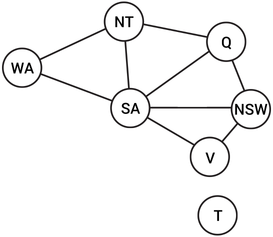
Can find each object owned by Eeyore in constant time and then check if it is a jar of hunny.

But what if Eeyore owns many objects but very few jars?

Conjunct Ordering Better (cost-wise) to find all jars of hunny first and then check whether they are owned by Eeyore.

Optimal ordering is NP-hard. Heuristics available: MRV from CSP if each conjunct is viewed as a constraint on its variables.

HARD MATCHING EXAMPLE



$\text{Diff}(WA, NT) \wedge \text{Diff}(WA, SA) \wedge \text{Diff}(NT, Q) \wedge \text{Diff}(NT, SA) \wedge \text{Diff}(Q, NSW) \wedge \text{Diff}(Q, SA) \wedge \text{Diff}(NSW, V) \wedge \text{Diff}(NSW, SA) \wedge \text{Diff}(V, SA) \to \text{Colourable}$

$\text{Diff}(Red, Blue)$, $\text{Diff}(Red, Black)$

$\text{Diff}(Black, Red)$, $\text{Diff}(Black, Blue)$

$\text{Diff}(Blue, Red)$, $\text{Diff}(Blue, Black)$

Every finite domain CSP can be expressed as a single definite clause + ground facts.

Colourable is inferred iff the CSP has a solution.

CSPs include 3SAT as a special case, hence matching is NP-hard.

12.c

Backward chaining

BACKWARD CHAINING · ALGORITHM

A function that returns multiple times, each time giving one possible result

function FOL-BC-ASK(KB, $query$) **returns** a generator of substitutions
  return FOL-BC-OR(KB, $query$, { })

Fetch rules that might unify

generator FOL-BC-OR(KB, $goal$, θ) **yields** a substitution
  for each rule ($lhs \Rightarrow rhs$) in FETCH-RULES-FOR-GOAL(KB, $goal$) **do**
    (lhs, rhs) $\leftarrow$ STANDARDIZE-VARIABLES((lhs, rhs))
    for each θ' **in** FOL-BC-AND(KB, lhs, UNIFY(rhs, $goal$, θ)) **do**
      yield θ'

Renaming of variables to avoid name clashes

generator FOL-BC-AND(KB, $goals$, θ) **yields** a substitution
  if $\theta = failure$ **then return**
  else if LENGTH($goals$) = 0 **then yield** θ
  else do
    $first$,$rest$ $\leftarrow$ FIRST($goals$), REST($goals$)
    for each θ' **in** FOL-BC-OR(KB, SUBST(θ, $first$), θ) **do**
      for each θ'' **in** FOL-BC-AND(KB, $rest$, θ') **do**
        yield θ''

BACKWARD CHAINING · EXAMPLE

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J})$ and $\text{Hunny}(\text{J})$

$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$

$\boxed{\text{Generous(Pooh)}}$

BACKWARD CHAINING · EXAMPLE

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, J)$ and $\text{Hunny}(J)$

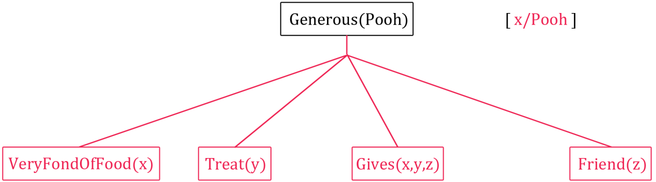
$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



BACKWARD CHAINING · EXAMPLE

VeryFondOfFood(x) $\wedge$ Treat(y) $\wedge$ Friend(z) $\wedge$ Gives(x, y, z) $\rightarrow$ Generous(x)

Owns(Eeyore, J) and Hunny(J)

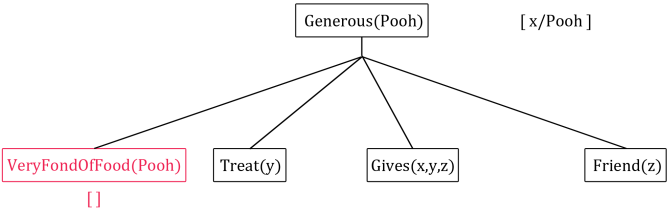
Hunny(x) $\wedge$ Owns(Eeyore, x) $\rightarrow$ Gives(Pooh, x, Eeyore)

Hunny(x) $\rightarrow$ Treat(x)

Resident(x, HundredAcreWood) $\rightarrow$ Friend(x)

Resident(Eeyore, HundredAcreWood)

VeryFondOfFood(Pooh)



BACKWARD CHAINING · EXAMPLE

VeryFondOfFood$(x) \wedge$ Treat$(y) \wedge$ Friend$(z) \wedge$ Gives$(x, y, z) \rightarrow$ Generous(x)

Owns(Eeyore, J) and Hunny(J)

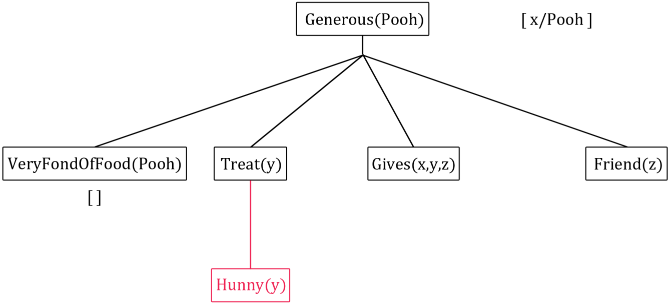
Hunny$(x) \wedge$ Owns(Eeyore, x) $\rightarrow$ Gives(Pooh, x, Eeyore)

Hunny$(x) \rightarrow$ Treat(x)

Resident(x, HundredAcreWood) $\rightarrow$ Friend(x)

Resident(Eeyore, HundredAcreWood)

VeryFondOfFood(Pooh)



BACKWARD CHAINING · EXAMPLE

$\text{VeryFondOfFood}(x) \wedge \text{Treat}(y) \wedge \text{Friend}(z) \wedge \text{Gives}(x, y, z) \rightarrow \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J})$ and $\text{Hunny}(\text{J})$

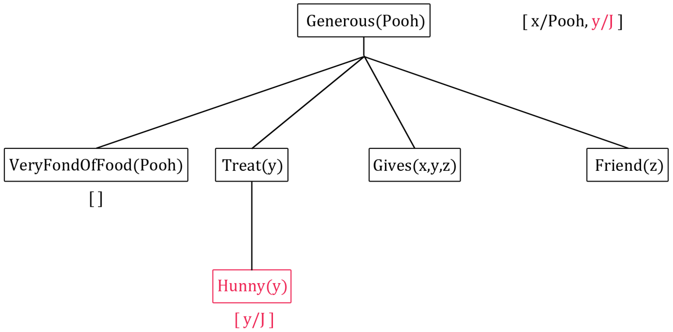
$\text{Hunny}(x) \wedge \text{Owns}(\text{Eeyore}, x) \rightarrow \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \rightarrow \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \rightarrow \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



BACKWARD CHAINING · EXAMPLE

VeryFondOfFood$(x) \land$ Treat$(y) \land$ Friend$(z) \land$ Gives$(x, y, z) \rightarrow$ Generous(x)

Owns(Eeyore, J) and Hunny(J)

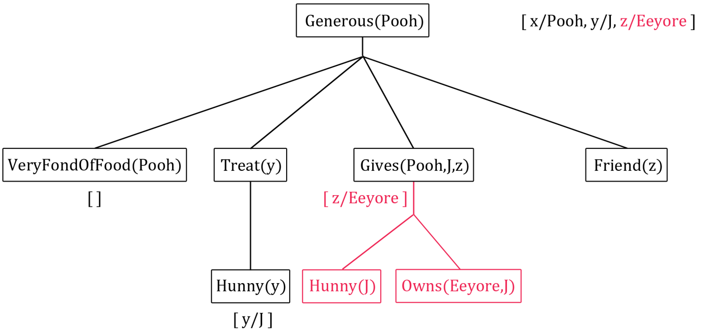
Hunny$(x) \land$ Owns(Eeyore, x) $\rightarrow$ Gives(Pooh, x, Eeyore)

Hunny$(x) \rightarrow$ Treat(x)

Resident(x, HundredAcreWood) $\rightarrow$ Friend(x)

Resident(Eeyore, HundredAcreWood)

VeryFondOfFood(Pooh)



BACKWARD CHAINING · EXAMPLE

$\text{VeryFondOfFood}(x) \land \text{Treat}(y) \land \text{Friend}(z) \land \text{Gives}(x, y, z) \to \text{Generous}(x)$

$\text{Owns}(\text{Eeyore}, \text{J})$ and $\text{Hunny}(\text{J})$

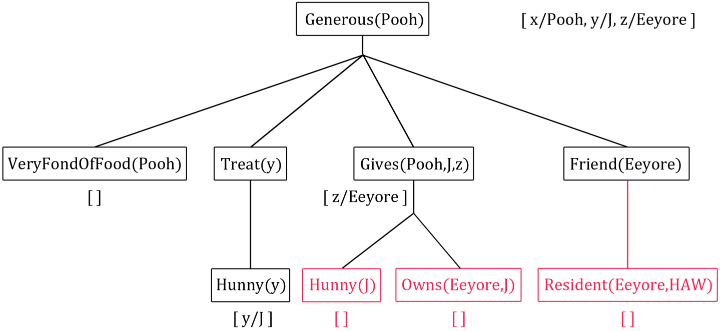
$\text{Hunny}(x) \land \text{Owns}(\text{Eeyore}, x) \to \text{Gives}(\text{Pooh}, x, \text{Eeyore})$

$\text{Hunny}(x) \to \text{Treat}(x)$

$\text{Resident}(x, \text{HundredAcreWood}) \to \text{Friend}(x)$

$\text{Resident}(\text{Eeyore}, \text{HundredAcreWood})$

$\text{VeryFondOfFood}(\text{Pooh})$



BACKWARD CHAINING · PROPERTIES

Depth-first recursive proof search: space is linear in size of proof.

Incomplete due to infinite loops.

partial fix by checking current goal against every goal on stack

Inefficient due to repeated subgoals (both success and failure).

fix using caching of previous results (extra space)

Widely used in **logic programming** languages.

"What's past is Prolog."

The Tempest, Act II, scene i