Inferenza nella logica dei predicati del primo ordine

Inferenza nella logica del primo ordine
Sostituzioni e quantificatori

Sost(ϑ, α) applica la sostituzione ϑ alla formula α

$\text{Sost}(\{x / Sam, y / Pam\}, Piace(x, y)\}) = Piace(Sam, Pam)$

Eliminazione degli universali

da $\forall x\ Piace(x,Gelato)$ sostituendo *{x/Ben}* si ha *Piace(Ben,Gelato)*

Eliminazione degli esistenziali

da $\exists x\ Uccide(x,Vittima)$ possiamo inferire $Uccide\ (Assa\sin o,Vittima)$

(Assassino è una qualsisasi costante che non compare nella formula)

Introduzione degli esistenziali

da *Piace(Gerry,Gelato)* possiamo inferire $\exists x Piace(x, Gelato)$

Inferenza nella logica del primo ordine

Dalle formule:

$Buffalo(Bob)$

$Pig(Pat)$

$\forall x, y\ Buffalo(x) \wedge Pig(y) \Rightarrow Faster(x, y)$

si inferisce la nuova formula: $Buffalo(Bob) \wedge Pig(Pat)$

e col Modus Ponens: $Faster(Bob, Pat)$

MODUS PONENS

se c'è una sostituzione ϑ tale che $\mathrm{Sost}(\vartheta, p_i) = \mathrm{Sost}(\vartheta, p_i^{'})\ \forall i$

$$\frac{p_1^{'}, p_2^{'}, \ldots, p_n^{'}, (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{\mathrm{Sost}(\vartheta, q)}$$

$p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q$ è una formula di Horn (un solo letterale positivo in una disgiunzione)

cioè: $\neg p_1 \vee \neg p_2 \neg \vee \ldots \neg \vee\ p_n \vee q$

Inferenza nella logica del primo ordine
UNIFICAZIONE

La funzione UNIFICA prende due formule atomiche *p* e *q* e restitusce una sostituzione che renda *p* e *q* *uguali.*

$$\text{UNIFICA}(p,q)=\vartheta \text{ dove } \text{Sost}(\vartheta, p)=\text{Sost}(\vartheta,q)$$

esempi :

$\text{UNIFICA}(Conosce(John,x),Conosce(John,Jane)) = \{x/Jane\}$

$\text{UNIFICA}(Conosce(John,x),Conosce(y,Leonid)) = \{x/Leonid, y/John\}$

$\text{UNIFICA}(Conosce(John,x),Conosce(y,Madre(y))) = \{y/John, x/Madre(John)\}$

$\text{UNIFICA}(Conosce(John,x),Conosce(x,Jane)) = fail$

Generalized Modus Ponens (GMP)

$$\frac{p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}',\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\sigma} \qquad \text{where } p_i{}'\sigma = p_i\sigma \text{ for all } i$$

$$\begin{aligned}
\text{E.g. } p_1{}' &= \text{Faster(Bob,Pat)} \\
p_2{}' &= \text{Faster(Pat,Steve)} \\
p_1 \wedge p_2 \ \Rightarrow\ q \ &= \ Faster(x,y) \wedge Faster(y,z) \ \Rightarrow\ Faster(x,z) \\
\sigma &= \{x/Bob, y/Pat, z/Steve\} \\
q\sigma &= Faster(Bob, Steve)
\end{aligned}$$

GMP used with KB of <u>definite clauses</u> (*exactly* one positive literal):
either a single atomic sentence or

(conjunction of atomic sentences) $\Rightarrow$ (atomic sentence)

All variables assumed universally quantified

Forward chaining

When a new fact p is added to the KB
for each rule such that p unifies with a premise
if the other premises are <u>known</u>
then add the conclusion to the KB and continue chaining

Forward chaining is <u>data-driven</u>
e.g., inferring properties and categories from percepts

Forward chaining example

Add facts 1, 2, 3, 4, 5, 7 in turn.
Number in [] = unification literal; $\surd$ indicates rule firing

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

Backward chaining

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

(Some added complications in keeping track of the unifiers)

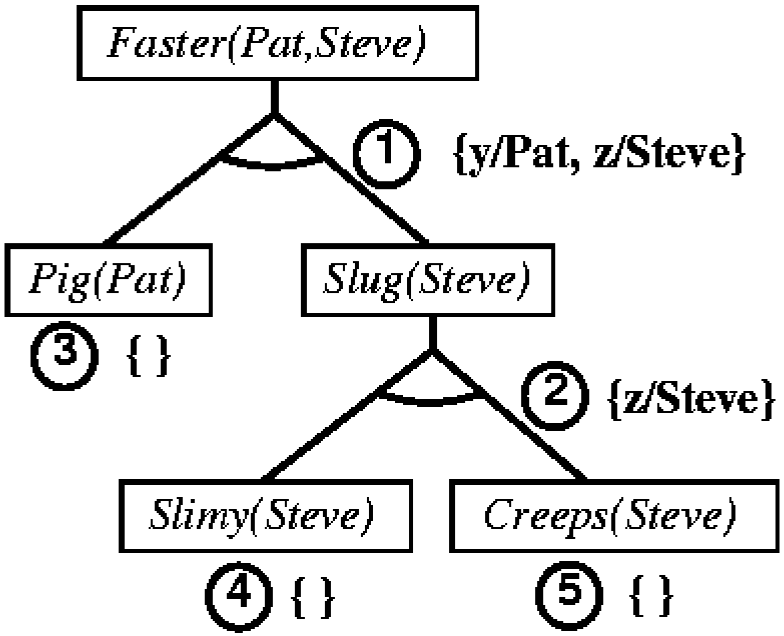
(More complications help to avoid infinite loops)

Two versions: find <u>any</u> solution, find <u>all</u> solutions

Backward chaining is the basis for <u>logic programming</u>, e.g., Prolog

Backward chaining example

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. $Creeps(Steve)$



Completeness in FOL

Procedure i is complete if and only if

$$KB \vdash_i \alpha \quad \text{whenever} \quad KB \models \alpha$$

Forward and backward chaining are <u>complete for Horn KBs</u> but incomplete for general first-order logic

E.g., from

$$PhD(x) \Rightarrow HighlyQualified(x)$$
$$\neg PhD(x) \Rightarrow EarlyEarnings(x)$$
$$HighlyQualified(x) \Rightarrow Rich(x)$$
$$EarlyEarnings(x) \Rightarrow Rich(x)$$

should be able to infer $Rich(Me)$, but FC/BC won't do it

Does a complete algorithm exist?

Inferenza nella logica del primo ordine

A brief history of reasoning

450B.C.	Stoics	propositional logic, inference (maybe)
322B.C.	Aristotle	"syllogisms" (inference rules), quantifiers
1565	Cardano	probability theory (propositional logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	$\exists$ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL (reduce to propositional)
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	"practical" algorithm for propositional logic
1965	Robinson	"practical" algorithm for FOL—resolution

Inferenza nella logica del primo ordine

Resolution

Entailment in first-order logic is only <u>semidecidable</u>:
can find a proof of α if $KB \models \alpha$
cannot always prove that $KB \not\models \alpha$

Cf. Halting Problem: proof procedure may be about to terminate with success or failure, or may go on for ever

Resolution is a <u>refutation</u> procedure:
to prove $KB \models \alpha$, show that $KB \wedge \neg\alpha$ is unsatisfiable

Resolution uses KB, $\neg\alpha$ in CNF (conjunction of clauses)

Resolution inference rule combines two clauses to make a new one:

$$\begin{tikzcd}[column sep=small] C_1 \arrow[dr] & & C_2 \arrow[dl] \\ & C & \end{tikzcd}$$

Inference continues until an <u>empty clause</u> is derived (contradiction)

Resolution inference rule

Basic propositional version:

$$\frac{\alpha \vee \beta, \ \ \neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \text{or equivalently} \quad \frac{\neg\alpha \ \Rightarrow \ \beta, \ \ \beta \ \Rightarrow \ \gamma}{\neg\alpha \ \Rightarrow \ \gamma}$$

Full first-order version:

$$\frac{\begin{array}{c} p_1 \vee \ldots \ p_j \ \ldots \vee p_m, \\ q_1 \vee \ldots \ q_k \ \ldots \vee q_n \end{array}}{(p_1 \vee \ldots \ p_{j-1} \vee p_{j+1} \ \ldots p_m \vee q_1 \ldots \ q_{k-1} \vee q_{k+1} \ \ldots \vee q_n)\sigma}$$

where $p_j\sigma = \neg q_k\sigma$

For example,

$$\frac{\begin{array}{l} \neg Rich(x) \vee Unhappy(x) \\ Rich(Me) \end{array}}{Unhappy(Me)}$$

with $\sigma = \{x/Me\}$

Conjunctive Normal Form

Literal = (possibly negated) atomic sentence, e.g., $\neg Rich(Me)$

Clause = disjunction of literals, e.g., $\neg Rich(Me) \lor Unhappy(Me)$

The KB is a conjunction of clauses

Any FOL KB can be converted to CNF as follows:

1. Replace $P \Rightarrow Q$ by $\neg P \lor Q$
2. Move $\neg$ inwards, e.g., $\neg \forall x\, P$ becomes $\exists x\, \neg P$
3. Standardize variables apart, e.g., $\forall x\, P \lor \exists x\, Q$ becomes $\forall x\, P \lor \exists y\, Q$
4. Move quantifiers left in order, e.g., $\forall x\, P \lor \exists x\, Q$ becomes $\forall x \exists y\, P \lor Q$
5. Eliminate $\exists$ by Skolemization (next slide)
6. Drop universal quantifiers
7. Distribute $\land$ over $\lor$, e.g., $(P \land Q) \lor R$ becomes $(P \lor Q) \land (P \lor R)$

Skolemization

$\exists x\, Rich(x)$ becomes $Rich(G1)$ where $G1$ is a new "Skolem constant"

$\exists\, k \;\; \frac{d}{dy}(k^y) = k^y$ becomes $\frac{d}{dy}(e^y) = e^y$

More tricky when $\exists$ is inside $\forall$

E.g., "Everyone has a heart"

$$\forall\, x \;\; Person(x) \;\Rightarrow\; \exists\, y \;\; Heart(y) \wedge Has(x, y)$$

Incorrect:

$$\forall\, x \;\; Person(x) \;\Rightarrow\; Heart(H1) \wedge Has(x, H1)$$

Correct:

$$\forall\, x \;\; Person(x) \;\Rightarrow\; Heart(H(x)) \wedge Has(x, H(x))$$

where H is a new symbol ("Skolem function")

Skolem function arguments: all enclosing universally quantified variables

Resolution proof

To prove α:

- negate it
- convert to CNF
- add to CNF KB
- infer contradiction

E.g., to prove $Rich(me)$, add $\neg Rich(me)$ to the CNF KB

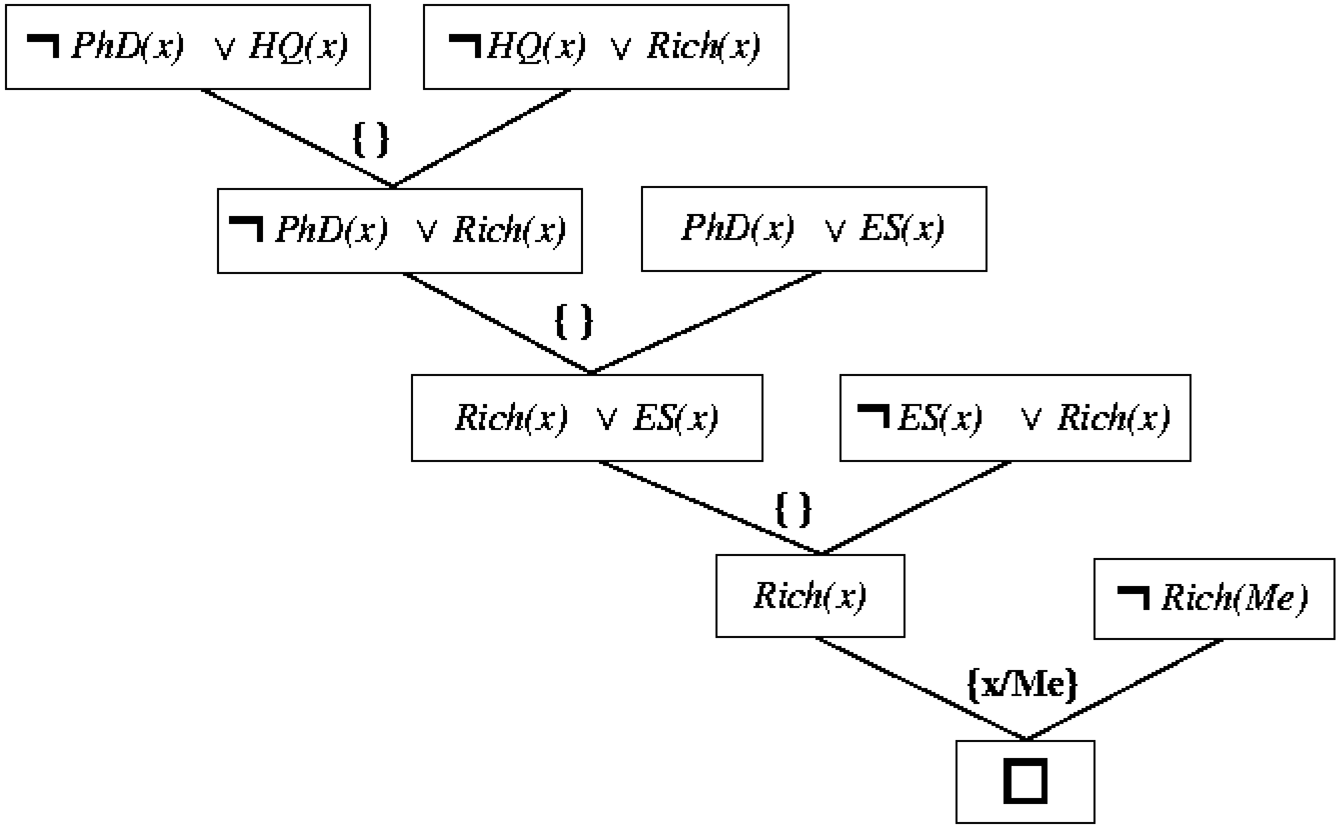
$\neg PhD(x) \vee HighlyQualified(x)$
$PhD(x) \vee EarlyEarnings(x)$
$\neg HighlyQualified(x) \vee Rich(x)$
$\neg EarlyEarnings(x) \vee Rich(x)$

Refutazione (dimostrazione per assurdo)

$$(KB \wedge \neg P \Rightarrow False) \Leftrightarrow (KB \Rightarrow P)$$

Inferenza nella logica del primo ordine

Resolution proof



Inferenza nella logica del primo ordine

Si consideri il problema *"se Marcia va ovunque vada John, e John è a scuola, dovè Marcia?"*

Assiomi :

$(\forall y)AT(John, y) \Rightarrow AT(Marcia, y)$

$AT(John, school)$

Domanda "*Dov' è Marcia ?*" $\quad (\exists x)AT(Marcia, x)$

Negazione della domanda : $\neg(\exists x)AT(Marcia, x)$

Che si può scrivere come : $(\forall x)\neg AT(Marcia, x)$

In forma a clausole :

$\neg AT(John, y) \vee AT(Marcia, y)$

$AT(John, school)$

$\neg AT(Marcia, x)$

Inferenza nella logica del primo ordine

$\neg AT(Marcia, x)$ $\qquad\qquad$ $\neg AT(John, y) \vee AT(Marcia, y)$

$\{y/x\}$

$\neg AT(John, x)$ $\qquad\qquad$ $AT(John, school)$

$\{x/school\}$

$\square$

Sostituzione di risposta: $\{x/school\}$

Sostituendo nella domanda: $AT(Marcia, x)$