First Order Inference

Inference Rules in FOL

- PL inference rules apply in FOL as well.
- Universal Elim: If $\forall x\, P(x)$ is true, then $P(c)$ is true for any constant c in the domain.
- Existential Intro: If $P(c)$ is true, then $\exists x\, P(x)$ is true.
- Existential Elim: If $\exists x\, P(x)$ is true, then $P(c)$ is true for some constant c not appearing in any other sentence (Skolem constant).

MONTANA STATE UNIVERSITY | Norm Asbjornson College of Engineering

Inference Rules in FOL

- Generalized Modus Ponens
  - Combines AND-Intro, Univeral-Elim, and Modus Ponens
  - Ex. From $P(c)$, $Q(c)$, and $\forall x\ (P(x) \wedge Q(x)) \Rightarrow R(x)$ derive $R(c)$.
  - Let subst(θ,α) denote substitutions resulting from applying substitution list θ to sentence α.
  - Given atomic sentences $P_1,\ldots,P_n$ and implication $(Q_1 \wedge \ldots \wedge Q_n) \Rightarrow R$ and subst(θ,P_i) = subst(θ,Q_i), derive subst(θ,R).

Inference in FOL

- Recall the Generalized Modus Ponens inference rule.
- This rule requires the ability to find substitutions allowing “facts” to match left-hand-sides of rules.
- This matching process is called “unification.”

MONTANA STATE UNIVERSITY | NORM ASBJORNSON College of ENGINEERING

Unification

- `unify(p,q) = θ`,
  - where `subst(θ,p)= subst(θ,q)`
  - `p` and `q` are sentences to be unified.
- Examples:
  - `unify(P(a,x),P(a,b))`
  - `unify(P(a,x),P(y,b))`
  - `unify(P(a,x),P(y,f(y))`
  - `unify(P(a,x),P(x,b))`

55

Unification

- `unify(p,q) = θ`,
  - where `subst(θ,p)= subst(θ,q)`
  - `p` and `q` are sentences to be unified.
- Examples:
  - `unify(P(a,x),P(a,b)) = {x/b}`
  - `unify(P(a,x),P(y,b)) = {x/b,y/a}`
  - `unify(P(a,x),P(y,f(y)) = {y/a,x/f(a)}`
  - `unify(P(a,x),P(x,b)) = FAIL`

MONTANA STATE UNIVERSITY | NORM ASBJORNSON College of ENGINEERING

56

Unification

- To avoid last problem, standardize apart variables.
- Replace first `x` with x_1 and second `x` with x_2.
- `unify(P(a,`x_1`),P(`x_2`,b)) = {`x_1`/b,`x_2`/a}`
- Not everything can be unified
  - `unify(P(a,b),P(x,x)) = FAIL`
  - `unify(P(a,f(a)),P(x,b)) = FAIL`
  - `unify(P(x),P(P(x))) = FAIL` (need "occurs check")

MONTANA STATE UNIVERSITY | NORM ASBJORNSON College of ENGINEERING

57

Unification Algorithm

function UNIFY(*x*, *y*, θ) **returns** a substitution making *x* and *y* identical
  inputs: *x*, a variable, constant, list, or compound
    y, a variable, constant, list, or compound
    θ, the substitution built up so far (optional, defaults to empty)

  if θ = *failure* **then return** *failure*
  else if *x* = *y* **then return** θ
  else if VARIABLE?(*x*) **then return** UNIFY-VAR(*x*, *y*, θ)
  else if VARIABLE?(*y*) **then return** UNIFY-VAR(*y*, *x*, θ)
  else if COMPOUND?(*x*) **and** COMPOUND?(*y*) **then**
    return UNIFY(ARGS[*x*],ARGS[*y*],UNIFY(OP[*x*],OP[*y*],θ))
  else if LIST?(*x*) **and** LIST?(*y*) **then**
    return UNIFY(REST[*x*],REST[*y*],UNIFY(FIRST[*x*],FIRST[*y*],θ))
  else return *failure*

MONTANA STATE UNIVERSITY | NORM ASBJORNSON College of ENGINEERING

58

Unification Algorithm

function UNIFY-VAR(*var*, *x*, θ) **returns** a substitution
inputs: *var*, a variable
x, any expression
θ, the substitution built up so far

if $\{var\ /\ val\} \in \theta$ **then return** UNIFY(*val*, *x*, θ)
else if $\{x\ /\ val\} \in \theta$ **then return** UNIFY(*var*, *val*, θ)
else if OCCUR-CHECK?(*var*, *x*) **then return** *failure*
else return $\theta + \{var\ /\ x\}$

Sample Proof

- Given:
  - Bob is a buffalo.
  - Pat is a pig.
  - Buffaloes outrun pigs.
- Prove:
  - Bob outruns Pat.

MONTANA STATE UNIVERSITY | NORM ASBJORNSON College of ENGINEERING

Sample Proof

- Convert premises to FOL.
  - *Buffalo*(*Bob*)
  - *Pig*(*Pat*)
  - $\forall x,y$ *Buffalo*(x) $\wedge$ *Pig*(y) $\Rightarrow$ *Faster*(x,y)
- Convert goal to FOL.
  - *Faster*(*Bob*,*Pat*)

61

Sample Proof

- Proceed with proof.
  - *Buffalo*(*Bob*) $\wedge$ *Pig*(*Pat*), AND Intro.
  - (*Buffalo*(*Bob*) $\wedge$ *Pig*(*Pat*)) $\Rightarrow$ *Faster*(*Bob*,*Pat*), Universal Elimination {x/*Bob*, y/*Pat*}
  - *Faster*(*Bob*,*Pat*), Modus Ponens   Q.E.D.

62