

**Class 6:
Indexing; Unify; Nothing New
in Semantic Nets, etc.; Guess
Who?; Planning**

Selmer Bringsjord

June 3, 1999

Logistics

- Make sure you're using our web site!
- Project 2 out:
 - up to and including 18.27
 - Supererogatory options:

- Project 3: (p1) done
 - Advanced options:
 - * grad students pick project for group that relates to your company (run by me)
 - * simple version of Hyperproof using a TP
 - * :

- Why Lisp? — we haven't talked about this, and we won't!
 - Built-in support for lists
 - Automatic storage management
 - Dynamic typing
 - First-class functions
 - Uniform syntax
 - Interactive environment
 - extensibility

- Answers to the Midterm: later! *Ralph*
- Second offer of automatic A (recall Collatz's problem)
 - First: infinitude *Hyperproof*
 - Formal diagnosis:
 - * so there is a set Γ of first-order formulas such that $\mathcal{I} \models \Gamma$ iff \mathcal{I} is infinite
 - What about finitude? Can it be expressed in FOL? I.e., is there a set that can play an analogous role for finitude?

- Indexing ...
- The unification algorithm ...
- Frame systems and semantic networks are just FOL ...
- You need to learn the foundation, FOL, inside and out!

- Planning as simply theorem proving
- initial state:
 - $\alpha_1[s_0] \wedge \alpha_2[s_0] \dots \alpha_n[s_0]$
- operators as successor-state axioms:
 - $\forall a, s \text{ Have}(Milk, \text{Result}(a, s)) \Leftrightarrow$
 - $((a = \text{Buy}(milk) \wedge \text{At}(\text{Supermarket}, s))$
 - $\vee (\text{Have}(Milk, s) \wedge a \neq \text{Drop}(Milk)))$
- Given this, simply:
 - $\vdash \exists p G(\text{Result}'(p, s_0))?$

- Next: more efficient representation schemes & POP

Key	Positive	Negative	Conclusion	Premise
<i>Brother</i>	<i>Brother(Richard, John)</i> <i>Brother(Ted, Jack)</i> <i>Brother(Jack, Bobbie)</i>	\neg <i>Brother(Ann, Sam)</i>	$Brother(x, y) \wedge Male(y)$ $\Rightarrow Brother(y, x)$	$Brother(x, y) \wedge Male(y)$ $\Rightarrow Brother(y, x)$ $Brother(x, y) \Rightarrow Male(x)$
<i>Male</i>	<i>Male(Jack)</i> <i>Male(Ted)</i> ...	\neg <i>Male(Ann)</i> ...	$Brother(x, y) \Rightarrow Male(x)$	$Brother(x, y) \wedge Male(y)$ $\Rightarrow Brother(y, x)$

Figure 1: Table for Indexing.

function UNIFY(x, y) **returns** a substitution to make x and y identical, if possible

UNIFY-INTERNAL($x, y, \{\}$)

function UNIFY-INTERNAL(x, y, θ) **returns** a substitution to make x and y identical (given θ)

inputs: x , a variable, constant, list, or compound
 y , a variable, constant, list, or compound
 θ , the substitution built up so far

if $\theta = \text{failure}$ **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-INTERNAL(ARGS[x], ARGS[y], UNIFY-INTERNAL(OP[x], OP[y], θ))

else if LIST?(x) **and** LIST?(y) **then**

return UNIFY-INTERNAL(REST[x], REST[y], UNIFY-INTERNAL(FIRST[x], FIRST[y], θ))

else return failure

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-INTERNAL(val, x, θ)

else if $\{x/val\} \in \theta$

then return UNIFY-INTERNAL(var, val, θ)

else if var occurs anywhere in x /* occur-check */

then return failure

else return add $\{x/var\}$ to θ

Figure 2: Unification Algorithm.

```

function SIMPLE-PLANNING-AGENT(percept) returns an action
  static: KB, a knowledge base (includes action descriptions)
           p, a plan, initially NoPlan
           t, a counter, initially 0, indicating time
  local variables: G, a goal
                     current, a current state description

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  current ← STATE-DESCRIPTION(KB, t)
  if p = NoPlan then
    G ← ASK(KB, MAKE-GOAL-QUERY(t))
    p ← IDEAL-PLANNER(current, G, KB)
  if p = NoPlan or p is empty then action ← NoOp
  else
    action ← FIRST(p)
    p ← REST(p)
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action

```

Figure 3: Simple Planning Agent.