Massachusetts Institute of Technology

6.038 Artificial Intelligence in Practice

Assignment 3: Thursday, February 20, 1997

1 Objective

The objective of this assignment is to give you experience in using a "practical" rule-based system and a fairly advanced theorem prover for first-order logic.

2 Files

You can find the `clips` software required for this assignment in `/mit/6.038/bin`; it has been compiled for Sun and SGI workstations.

Make sure that you `add 6.038` when starting to work on this assignment. This will put the correct bin directory on your PATH.

The theorem prover runs in Lucid Lisp (I recommend that you run the following commands in a shell under Emacs). Lucid Lisp only runs on Athena Suns.

```
add lucidcl
lisp
(lcl:cd "/mit/6.038/dtp/src")
(load "dtp") ;; loads source code
(in-package 'dtp)
(lcl:cd "")
(dtp-load "/mit/<your directory>/blocks.dtp") ; load your theory
(setq *theory* 'blocks)
(setq *trace* '(:conjuct-proofs :subgoals)) ; setup tracing
(prove '(on green yellow)) ; for example
(show-proof-graph *last-proof*)
```

Now you're ready to read in your axioms and try to do proofs (see below).

The other files mentioned below can be found in `/mit/6.038/assign-3/`.

3 Part 1: Rules in Clips

Look at the file `animal.clp`. Explain the `defrule`s in this file. How does this system work? What is the relation of this system of rules to backward chaining?

Look at the file `auto.clp`. Write a set of rules that emulate the performance of the auto rules but using the approach in the animal rules. Compare the original auto rules to these rules? Which set do you think is better? Why?

4 Part 2: Therem proving with DTP

The following facts (due to Mike Genesreth at Stanford) are given to you:

```
The red block is on the green block.
The green block is somewhere ABOVE the blue block.
The green block is NOT on the blue block.
The yellow block is on the green block or on the blue block.
The blue block is on SOME other block.
```

We want to prove that the following facts follow from these given facts (and some more general axioms).

```
The yellow block is on the blue block.
The blue block is on the black block.
The green block is on the yellow block.
The black block is on the table.
```

The informal "proofs" of these facts are as follows:

- We are told that the yellow is on the green or the blue. Since the red is on the green, the yellow must be on the blue.
- We are told that the blue is on some block. It cannot be the green, since the green is above the blue. It cannot be the red since the red is on the green. It cannot be the yellow since the yellow is on the blue. So, it must be the black.
- We are told that the green is above the blue. Therefore, it must be on some block. It cannot be on the red since the red is on the green. Obviously, the green cannot be on itself. Since the yellow is on the blue, we know the green cannot be on the blue. Since the blue is on the black, the green cannot be on the black. Therefore, it must be on the yellow.
- All other blocks are above the black block, so the black block must be on the table.

Specify the complete set of axioms needed to prove these facts formally using DTP. A start in that direction can be found in the file `blocks.dtp`. In particular, it has the initial facts and a couple of axioms to fake equality tests. You should add your axioms to this theory. All the variables in the axioms are universally quantified, so if you need an existentially quantified variable, you need to use a function (that is, Skolemize). You do not need to do the full conversion to clause form (although you'll see that when the system does it, it ends up with some wasteful statements). Do `(show '<theory name>)` to see the statements in the theory.

This is a lot harder than it seems (or should be). You should try to sketch the proofs out formally and see what axioms you need before you run off trying things with the theorem prover. Try to do the first proof, using the fewest axioms, first so you get used to the system. By the way, every time you change the theory in the file, you need to reload it into the Lisp (using `(dtp-load "/mit/<your directory>/blocks.dtp")`).

It is useful to see what the theorem prover is up to during its search, that is what `(setq *trace* '(:conjuct-proofs :subgoals))` does. To cut down on printing, you can do `(setq *trace* '(:subgoals))` instead.

What do you conclude about the practicality of theorem provers from this exercise? Hand in the final file `blocks.dtp`. DON'T hand in the proofs...

Do not spend too much time on this problem (unless you're enjoying it). See how much you can do in a reasonable amount of time. I want you to get a feeling for the mechanics of doing this and how hard it can be.