CPSC 322
Introduction to Artificial Intelligence

October 13, 2004

Problem 1a

1a. (10 points) Let's see how well you really understand this interpretation stuff. You know of course from your reading and from lecture that once you've completely and thoroughly specified your intended interpretation for your chosen domain, you then write, as definite clauses, those statements that are true in your intended interpretation. Those clauses are called axioms. It seems reasonable then that given only a set of axioms (i.e., definite clauses) that make up the facts of a knowledge base, you should be able to work backwards from that information and completely and thoroughly specify the intended interpretation. Well actually, you'll need a little more information that what's contained in the axioms. Read on.

Here's a set of axioms in definite clause form:

```
cries(stanley).
crushes_hat(ollie,stanley).
crushes_hat(stanley,ollie).
```

Assume that `stanley` and `ollie` are the only constants, and that Stanley' and Ollie' are the only individuals worth worrying about in the chosen domain. Assume that `stanley` maps onto Stanley' and `ollie` maps onto Ollie'. Assume also that `cries` and `crushes_hat` are the only predicates. (And of course assume that the axioms are correct, no mistakes were made, and the axioms represent the only true statements in the interpretation.) In the space below, give the complete specification for the intended interpretation that gives rise to the axioms above. You can use the words *pi* and *phi* or their symbolic equivalents, ϕ and π.

Problem 1a

D = {Stanley', Ollie'}

ϕ(stanley) = Stanley'
ϕ(ollie) = Ollie'

π(cries)(Stanley') = True
π(crushes_hat)(Stanley',Ollie') = True
π(crushes_hat)(Ollie',Stanley') = True

π(cries)(Ollie') = False
π(crushes_hat)(Stanley',Stanley') = False
π(crushes_hat)(Ollie',Ollie') = False

Problem 1b

1b. (10 points): Given the facts and assumptions of Problem 1a, how many different interpretations are possible? Show enough math to convince me that you didn't just pull a number out of the air. (Hint: the number of interpretations possible is not nearly as large as the number of interpretations for the similar problem in homework assignment 1.)

Problem 1b

D = {Stanley', Ollie'}

ϕ(stanley) = Stanley' or Ollie'	2
ϕ(ollie) = Ollie' or Stanley'	2
π(cries)(Stanley') = True or False	2
π(crushes_hat)(Stanley',Ollie') = True or False	2
π(crushes_hat)(Ollie',Stanley') = True or False	2
π(cries)(Ollie') = False or True	2
π(crushes_hat)(Stanley',Stanley') = False or True	2
π(crushes_hat)(Ollie',Ollie') = False or True	2

$2^8 = 256$

Problem 2

2. (20 points): Let's say for the sake of argument that we all believe that, someday, a reasoning and representation system like the ones we've been working with in class could be intelligent. (Well, it would be a lot more complex than the ones we've been working with, but it would employ the same principles.) This belief is based on a handful of assumptions. In the space below, explain just two of those assumptions. (Note: this isn't about the textbook's simplifying assumptions like "The environment is static" that are relaxed or eliminated completely later in the book.) Once you've given those two assumptions, discuss the implications for artificial intelligence efforts if those assumptions ultimately turn out to be false.

AI assumes that what the brain does may be thought of at some level as some form of computation

The assumption above is probably valid

A physical symbol system has the necessary and sufficient means for intelligent behavior
(The Physical Symbol System Hypothesis)

These assumptions may not be valid

Any symbol manipulation can be carried out on a Turing machine.
(The Church-Turing Thesis)

Alternatives to symbols

Number crunching (e.g., language processing entirely by statistical analysis)

Distributed intelligence

- Lots of tiny “computers” of limited ability working in concert (e.g., ants in a colony, neurons in a brain)

Problem 3

3. (20 points): Here's a graphical representation of a really small maze:

```
< the drawing would be here >
```

And here are the corresponding facts in CILOG:

```
nowall(r11,r12).
nowall(r12,r13).
nowall(r11,r21).
nowall(r12,r22).
nowall(r21,r31).
nowall(r13,r23).
nowall(r31,r32).
nowall(r23,r33).
```

Now let's add a couple of rules:

```
path(X,Y) <- path(X,Z) & path(Z,Y).
path(X,Y) <- nowall(X,Y).
```

Prove the following theorem is true (i.e., show that the theorem can be derived from the given knowledge base):

```
path(r11,r32).
```

Problem 3

```
nowall(r11,r12).
nowall(r12,r13).
nowall(r11,r21).
nowall(r12,r22).
nowall(r21,r31).
nowall(r13,r23).
nowall(r31,r32).
nowall(r23,r33).

path(X,Y) <- path(X,Z) & path(Z,Y).
path(X,Y) <- nowall(X,Y).
```

Prove the following theorem is true (i.e., show that the theorem can be derived from the given knowledge base):

```
yes  <- path(r11,r32).
     <- path(r11,Z) & path(Z,r32).
     <- nowall(r11,Z) & path(Z,r32).
     <- nowall(r11,r21) & path(r21,r32).
     <- path(r21,r32).
     <- path(r21,Z) & path(Z,r32).
     <- nowall(r21,Z) & path(Z,r32).
     <- nowall(r21,r31) & path(r31,r32).
     <- path(r31,r32).
     <- nowall(r31,r32).
     <- .
```

Problem 4

4. (20 points): Using CILOG, write a relation

```
list_equal(L1,L2)
```

that is true if the list L1 is identical to list L2. The relation is false otherwise. Sample behavior follows:

```
cilog: ask list_equal([],[]).
Answer: list_equal([], []).
  [ok,more,how,help]: ok.
cilog: ask list_equal(a,a).
No. list_equal(a, a) doesn't follow from the knowledge base.
cilog: ask list_equal([a,b,c],[a,b,c]).
Answer: list_equal([a, b, c], [a, b, c]).
  [ok,more,how,help]: ok.
cilog: ask list_equal([a,b,c,d],X).
Answer: list_equal([a, b, c, d], [a, b, c, d]).
  [ok,more,how,help]: more.
No more answers.
cilog: ask list_equal([a,b],[a,b,c]).
No. list_equal([a, b], [a, b, c]) doesn't follow from the knowledge base.
cilog:
```

Problem 4

The most frequent correct answer:

```
list_equal([H|T1],[H|T2]) <- list_equal(T1,T2).
list_equal([],[]).
```

Other possibilities:

```
list_equal([H|T],[H|T]).
list_equal([],[]).

list_equal(L1,L2) <- append(L1,[],L2).
append([], A, A).
append([A|B], C, [A|D]) <- append(B, C, D).
```

Problem 5

5. (20 points). Assume the following relation written in CILOG:

```
glork([H1|[H2|T2]],[H1|T3]) <- glork(T2,T3).
glork([],[]).
```

Show what happens when CILOG is presented with the following query:

```
ask glork([a,b,c,d,e,f],X).
```

Give the result, and explain how `glork` works.

```
glork([a,b,c,d,e,f],[a,c,e]).
```

`glork` proves that the second list is the list composed of the first, third, fifth, etc., elements of the first list (assuming that the first list has an even number of elements). It does this by comparing the first element of the first list to the first element of the second list. If they're the same, the proof procedure recursively calls itself with new arguments: the new first list is the original first list without its first two elements, and the new second list is the original second list without its first element. Thus, the second, fourth, sixth, etc. elements of the first list are ignored or discarded. The base case is when the two lists are empty.