Computer Science 81, Spring 2007
Assignment 10
Due Mon. April 16
Computability

1. [50 points] Show that there is no algorithm for determining whether a set of predicate logic formulas is unsatisfiable.

   Suggestion: Start with the result that there is no algorithm for determining whether an arbitrary Turing machine halts on a given input. Show that for any Turing machine M and initial tape x, there is a set of clauses such that the set of clauses is unsatisfiable iff M halts on x.

   This could be established by using one 1-ary function symbol for each distinct tape symbol. The clauses would define the possible moves of the Turing machine. (Note that you will need to represent blank tape and tape expansion.) This is similar to the pegs puzzle example given in class. Clauses for it can be found on the course web page as Pegs Puzzle.

   Demonstrate your technique on a specific M and x by giving the corresponding clauses to Otter-λ and have it check unsatisfiability. Do this for both halting and non-halting cases. The examples to use are given by the two TM tables below. The tape alphabet is {**a**, **x**, **b**} where **b** represents blank. It is assumed the tape has some number of contiguous **a**'s and that the head is positioned at the rightmost **a**, if there is any. Program **double** is supposed to double the number of contiguous **a**'s, which it does by adding an **x** at the right end for each **a**, after replacing that **a** with an **x**. Once each **a** has produced two **x**'s, it converts all **x**'s back to **a**'s. If the head starts over a **b**, that indicates that there are no **a**'s to double. State s is the start state. Program **damaged** is a damaged version of **double**, which doesn't converge. In it, the symbols shown in parentheses in the table replace the other symbols. Show your work with input **baaaa**.

double				
Current state	Symbol read	Symbol written	Head moves	Next state
s	b	b	left	t
s	a	x	right	u
t	b	b	right	v
t	a	x	right	u
t	x	x	left	t
u	x	x	right	u
u	b	x (b)	left	t
v	x	a	right	v
v	b	b	left	e (s)

2. [20 points] Show that the following are primitive-recursive:

   a. The pairing function

      $p(x, y) = 2^x (2y + 1) - 1$

   b. The inverses of p, i.e. the functions L and R, such that
      $L(p(x, y)) = x$ and $R(p(x, y)) = y$.

3. [20 points] Show that if h and k are primitive recursive functions, so are f and g, defined by:

   $f(0) = a$, $g(0) = b$

   $f(n+1) = h(f(n), g(n), n)$
   $g(n+1) = k(f(n), g(n), n)$

4. [10 points] Show that the function f defined by:

   $f(0) = 1$
   $f(1) = 1$
   $f(n+2) = f(n) + f(n+1)$

   is primitive recursive.