COMP 455
Models of Languages and Computation
Spring 2011
A Turing Machine-Like Language

We present a high-level language for describing Turing machines. Programs in this language can be translated fairly easily into Turing machines. This language consists of the following kinds of statements:

1. Boolean conditions of the form "ts = a" where "ts" denotes the tape symbol scanned and "a" is an element of the tape alphabet.

2. Boolean combinations of Boolean conditions using the connectives "and," "or," and "not."

3. Executable statements of the following forms, all of which can have labels:

   - The statement "L," signifying moving to the left, and "R," signifying moving to the right.
   - "If" and "while" statements using Boolean conditions.
   - Statements of the form "write(a)" where "a" is a tape symbol.
   - Statements of the form "goto S" where "S" is a statement label.
   - "Halt(s)" statements, where "s" is a halting state of a Turing machine.

We can abbreviate "ts = a or ts = b or ts = c" by "ts in {a,b,c}," and we can also use abbreviations like "ts not in {a,b,c}." Statements can be grouped using { and } as in C.

An example program for searching to the right for a blank and then writing a "b" is:

```
{while not(ts=⊔) R}; write(b);
```

Such programs can be compiled by standard techniques into programs involving only the statements "L," "R,", "write(a)," "halt(q)," "goto S," and "if(ts=a)goto S." By considering each address as a state, these compiled programs can easily be translated into Turing machines.