CS 342 — Principles of Programming Languages May 17, 1996

Turing Machine Reference Manual

This document describes the turing machine language that will be used for this term's main programming assignments.

1 Syntax

1.1 Syntax of Programs and Input

The syntax of a turing machine program and the simulator input is given by the following extended BNF grammar.

```
⟨program and input⟩ ::= ⟨program⟩ ⟨tape⟩
⟨program⟩ ::= ( ⟨instruction⟩* )
⟨instruction⟩ ::= ( ⟨current state⟩ ⟨read char⟩ = ⟨write char⟩ ⟨direction⟩ ⟨next state⟩ )
⟨current state⟩ ::= ⟨state label⟩
⟨next state⟩ ::= ⟨state label⟩
⟨read char⟩ ::= ⟨simulated blank⟩ | ⟨printing char⟩
⟨write char⟩ ::= ⟨printing char⟩
⟨direction⟩ ::= left | right
```

1.2 Lexical conventions

White space (tabs, blanks, and newlines) may be used freely to surround tokens that appear in the above grammar. The parts of an instruction must be separated by white space.

The tokens appearing above are defined in the following extended BNF grammar. The simulated blank character is an underbar (_).

```
⟨tape⟩ ::= ⟨tape char⟩*
⟨tape char⟩ ::= ⟨simulated blank⟩ | ⟨printing char⟩
⟨simulated blank⟩ ::= _
⟨printing char⟩ ::= ⟨digit⟩ | a | A | b | B | ··· | z | Z
⟨state label⟩ ::= ⟨digit⟩ ⟨digit⟩*
⟨digit⟩ ::= 0 | 1 | 2 | ··· | 9
```

1.3 Examples

For example, the following is a complete program and its input

```
((0 r = w left 4)
 (0 x = y right 0)
)
this_is_the_input__TAPE_initially_wise_27a_
```

The minimal program and input is `()`.

2 Semantics

A $\langle$program$\rangle$ defines a transition function δ which is partial mapping from states and tape characters to states, printing characters, and a direction:

$$\delta : \langle \text{state label} \rangle \times \langle \text{tape char} \rangle \rightarrow \langle \text{state label} \rangle \times \langle \text{printing char} \rangle \times \langle \text{direction} \rangle$$

The function δ is derived from the $\langle$program$\rangle$ as follows. The simulator must ensure that no two $\langle$instruction$\rangle$s have the same $\langle$current state$\rangle$ and $\langle$read char$\rangle$[1]. Given that the program meets that condition, for each s_1 and r, $\delta(s_1, r) = (s_2, w, d)$ if and only if there is some instruction in the program of the form

$$(s_1 \;\; r \;\; \texttt{=} \;\; w \;\; d \;\; s_2)$$

that is, if there is an instruction in the program that has s_1 as its $\langle$current state$\rangle$, r as its $\langle$read char$\rangle$, w as its $\langle$write char$\rangle$, d as its $\langle$direction$\rangle$, and s_2 as its $\langle$next state$\rangle$; otherwise δ is undefined on s_1 and r.

A $\langle$tape$\rangle$ defines the initial input tape for the simulated turing machine. The initial tape consists of the characters in $\langle$tape$\rangle$, with the leftmost character of $\langle$tape$\rangle$ as the leftmost character of the input tape.

Let a *configuration* of a Turing machine be a tuple (q, α, i) where q is an integer (the current state), $\alpha \in \langle \text{tape char} \rangle^*$ (the tape), and i is an integer (the position of the machine's tape head). Let α_i denote the i^{th} character of α, counting from the left, with 1 as the first character.

The *initial configuration* of the simulator whose input has a program and the $\langle$tape$\rangle$ t is $(\texttt{0}, t, 1)$ if t is not empty; if t is empty, it is $(\texttt{0}, \texttt{\_}, 1)$.

The simulation of the turing machine program on its input tape proceeds as follows. Starting from the initial configuration, the simulator makes transitions as specified by δ (and described in more detail below). If, the simulation reaches a point where no more transitions are possible, then the simulation is finished by printing the nonblank portion of the final simulated tape (that is the α from the final configuration) on standard output.

In more detail, the transitions are defined as follows. Suppose the current configuration is (q, α, i) and δ is such that $\delta(q, \alpha_i) = (s_2, w, d)$. There are several cases:

- If d is `left` and i is 1, then no transition is possible. This is an error (trying to move to the left off the end of the tape).
- If d is `left` and i is not 1, then the next configuration is $(s_2, \beta, i - 1)$, where β is the same as α, except that $\beta_i = w$.
- If d is `right`, then the next configuration is $(s_2, \beta, i + 1)$, where β is the same as α, except that $\beta_i = w$ and if α only had i characters, then β_{i+1} is defined to be the simulated blank character (_).

[1] The assumption that the current state and read char uniquely determine the next configuration means that the simulator only allows deterministic turing machine programs.