Turing Machine Simulation Language

Isaac McGarvey
Joshua Gordon
Keerti Joshi
Snehit Prabhu

Overview

- What is a TM
- Basic language

  Read/Write, Moving along tape head, Control constructs,

  Arbitrary tape alphabet.

- Scope of Language
  - Single tape datastructure
  - Single pass compilation without look-ahead.

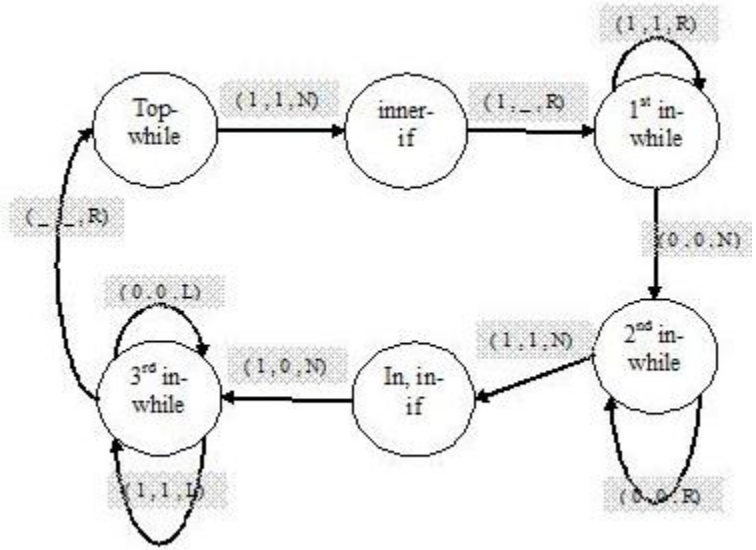
Overview

- Evolution of TMSL
  - Started with configuration file syndrome
  - Moved on to an ambitious high-level language plan
  - Converged on low-level scripting language with a tractable mapping from script to TM constructs.
- Limitations
  - No function definitions or code reusability.
  - No arithmetic.
  - No variables.

Language Overview

- Grammar:
  - Our programs are composed of symbols and statements
  - A symbol list
    - which specifies the characters which may be written to and read from the tape (in addition to the special blank character)
  - A statement list
    - Which specifies control flow and commands
    - **Statements** are generally of two types:
      - **Atomics** (e.g., left, right, write, exit)
      - **Composites** (e.g., if, while, until, unless)
    - Composites are simply statements which contain lists of statements within their definition, e.g.
      - UNLESS LPAREN symbol_list RPARENLBRACE stmt_list RBRACE

Writing a program: Unsigned subtraction



- 0,1 /*alphabet specification */
- while (1) {
- if (1) {
- write _
- right
- while (1) {
- right
- }

- Sample Input : 1111011_ _ _
- Output : _ _ 11000 _ _ _

- **Demo?**

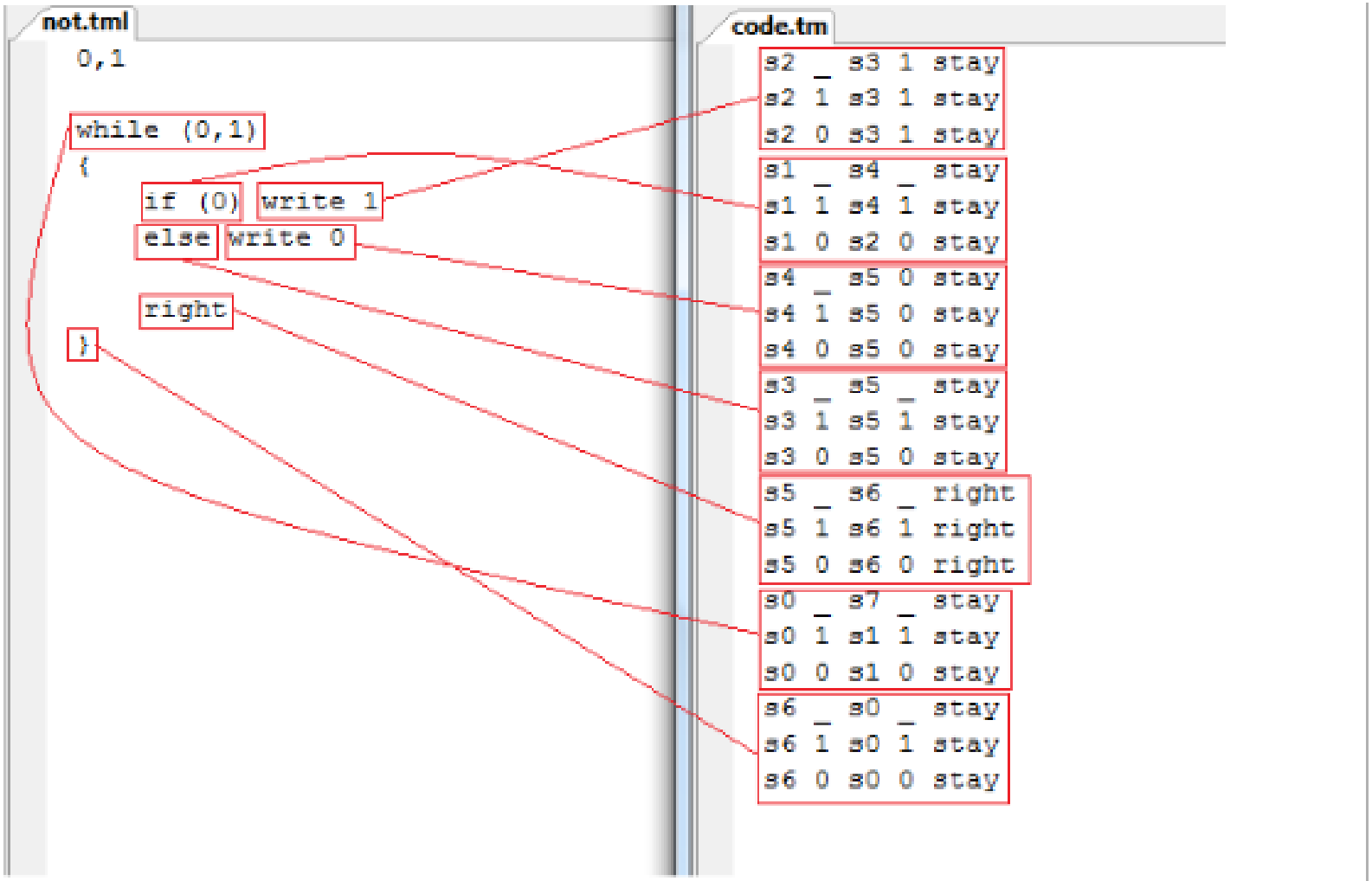
Implementation

- Machine Simulator
  - 2 scanners & parsers
  - Following transitions stored in list
  - Dynamically growing input tape

Implementation

- Compiler
  - Scanner & parser
  - AST types: statements, symbols
  - Code generation
    - Single pass: minimal semantic error checking
    - Translating statements into states and transitions
    - Bookkeeping: assigning state numbers to statements

Implementation



Summary and Lessons Learned

Language Design concepts

- Even out work through the semester
- Test cases (Regression suite)
- Good experience with the language
- Its fun to build your own compiler!