# Turing Machine Variants

Mark Greenstreet, CpSc 421, Term 1, 2008/09

- A Programmable Turing Machine
- Other variants
  - Multi-Tape Turing Machines
  - Non-Deterministic Turing Machines
  - Programming languages

# A Programmable Turing Machine

See slides for October 20.

# Multi-Tape Turing Machines

Consider a machine with $k$ tapes.

- Each tape, $i$, has its own alphabet, $\Gamma_i$.
- The machine has $k$ tape heads, one for each tape.
- At each step the machine:
  - reads the symbol under each head
  - based on these $k$ symbols, the machine:
    - writes a new symbol on each tape (it can write different symbols for different tapes);
    - moves each tape head one square to the left or right;
    - transitions to a new state.
  - Initially:
    - The input string is written on the first tape (followed by an infinite string of blanks).
    - The other strings are filled completely with blanks;
    - Each tape head is at the leftmost square of its tape.
  - The machine accepts, rejects or loops just like a regular Turing machine.
  - We could formalize all of this with tuples, but we won't.

# Simulating a Multi-Tape TM

Constructing a single-tape TM that simulates a multi-tape TM.

- Make an alphabet that combines the alphabets for the $k$ tapes: $\prod_{i=1}^{k} \Gamma_i$.
- But, we also need to keep track of the $k$ head locations.
  - We do this by marking squares.
  - Each symbol needs a marked, and an unmarked version.
  - For alphabet $\Gamma_i$, let $\Gamma_i'$ be the corresponding set of "marked" symbols.
  - Our tape alphabet will be $\prod_{i=1}^{k}(\Gamma_i \cup \Gamma_i')$.
- Likewise, the state of the single-tape TP will be a tuple
  - The first component holds the state of multi-tape TM.
  - The second component holds a state component for the simulation procedure described on the next slide.
  - The remaining $k$ components record the symbol under each tape head.

# Simulating a Multi-Tape TM (cont.)

At each step:

- Sweep the tape and remember the marked symbol for each tape.
- Based on these symbols and the current state of the multitape machine, determine:
  - The symbol to be written on each tape;
  - The direction to move each head;
  - The new state for the multi-tape machine.
- Sweep the tape again and update the symbols for each tape head.
  - Note that this means writing a symbol that keeps the components for the other tapes unchanged, and modifies the symbol for the tape with the marked symbol.
  - The markers also need to be moved according to the directions determined above.
- If the simulated multi-tape machine accepts, then the simulating machine accepts as well. Likewise for rejects.

# Multi-Tape Equivalence

- We've shown that any computation that can be performed by a multi-tape TM can be performed (e.g. simulated) by a single-tape TM.
  - In particular, if there is a multi-tape TM that recognizes (resp. decides) language $B$, then there is a single-tape that recognizes (resp. decides) $B$ as well.
- Every single-tape TM is just a special case of a multi-tape TM.
  - Thus, if there is a single-tape TM that recognizes (resp. decides) language $B$, then there is a multi-tape that recognizes (resp. decides) $B$ as well.
- $\therefore$ Single-tape and multi-tape TM's are equivalent in computational power.

# Non-Deterministic TMs

- Definition
- Showing equivalence with deterministic TMs
  - Sipser's method: enumerating the choices.
  - An alternative: simulating all of the branches.

# Equivalence with other models

- Programming languages
- Computers
- Other . . .

# This week

- Reading
  - October 22 (Today): *Sipser* 3.3.
  - October 24 (Friday): *Sipser* 4.1.
  - October 27 (Monday): *Sipser* 4.2.
  - October 29 (A week from today): *Sipser* 4.2 (continued).
- Homework
  - October 24 (a week from today): Homework 6 due; homework 7 goes out.