CS 3313
Foundations of Computing:

Computing Functions on a Turing Machine

http://gw-cs3313.github.io

1

Next..

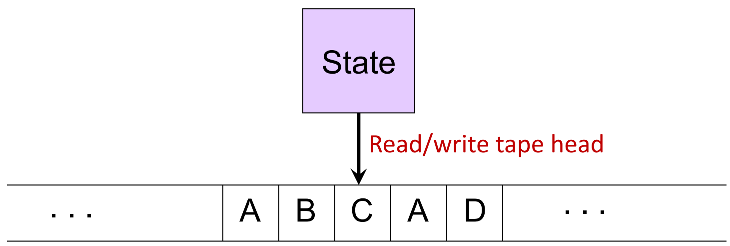
- Review: Turing Machine model
  - TM as an automaton
- Today: Computing functions on a Turing machine
  - Turing machine as a "computer"
  - First step is encoding the integer arguments to the TM
- ….TM "programming" techniques
  - Storage in the state (you've seen this) ✓
  - Checking/marking symbols ✓
  - Shifting over (skipping) tape symbols ✓
  - Subroutines…?
  - Multiple tracks on the tape

2

Turing Machine

Action: based on the (i) state and (ii) the tape symbol under the read/write head:

- (1) change state, (2) write a symbol back to the tape and (3) move the head (left or right) one location/cell on the tape.



Infinite tape with cells containing tape symbols chosen from a finite alphabet

3

Turing-Machine Formalism

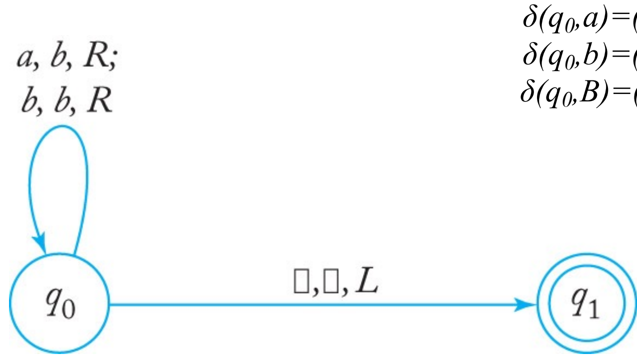
- A TM is described by:
  1. A finite set of *states* Q.
  2. An *input alphabet* Σ.
  3. A *tape alphabet* Γ (contains Σ).
  4. A *transition function* $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$
  5. A *start state* q_0 (in Q).
  6. A *blank symbol* B (or □) in $\Gamma - \Sigma$
     - All tape except for the input is blank initially.
  7. A set of *final states* $F \subseteq Q$

4

Transition Graphs for Turing Machines

- In JFLAP, you will see a transition graph for a TM
- In a Turing machine transition graph, each edge is labeled with three items:
  1. current tape symbol,
  2. new tape symbol, and
  3. direction of the head move

$\delta(q_0,a)=(q_0,b,R)$
$\delta(q_0,b)=(q_0,b,R)$
$\delta(q_0,B)=(q_1,B,L)$



5

Formal Definition of Moves: Instantaneous Description (ID)

- At any instant in time, the TM is in a state q, its tape head is reading some symbol Z, the string α is to the left of the tape head, and the string β is to the right of the tape head:

  this ID is denoted as $\alpha qZ\beta$
- Moves of a TM take TM from one ID to another..be defined using $ID_1 \vdash ID_2$ and $ID_1 \vdash^* ID_2$ to represent "in one move" and "in zero or more moves,"
- If $\delta(q, Z) = (p, Y, R)$, then $\alpha qZ\beta \vdash \alpha Yp\beta$
  - If Z is the blank B, then also $\alpha q \vdash \alpha Yp$
- If $\delta(q, Z) = (p, Y, L)$, then for any X, $\alpha XqZ\beta \vdash \alpha pXY\beta$
  - In addition, $qZ\beta \vdash pBY\beta$

6

6

Languages of a TM

- A TM defines a language by final state, as usual.
- $L(M) = \{w \mid q_0 w \vdash^* I$, where I is an ID with a final state$\}$.
  - TM halts in this configuration
  - Alternate definition: accepts as long as state is a final state
- A language that is accepted by a TM is called a *recursively enumerable language*
  - Strings than can be enumerated by a TM
  - TM may not halt on an input not in the language
- A language that is accepted by a TM that *halts on all inputs* is called a *recursive language*

7

7

Turing Machine as a transducer…i.e., as a computer of functions

- Definition:
- TM M starts with input string w and in ID $q_0 w$
- Let TM halt (in accepting) state q_f and y is string on the tape
  - $q_0 w \vdash_M^* q_f y$
- *Then,* the function computed by the TM M is $y = f(w)$

8

Turing Machine as a computer of functions: Encoding integers

- *If $q_0 w \vdash_M^* q_f y$ then M computes $f(w) = y$*
- We want to compute functions over integers: $q_0 \langle x \rangle \vdash_M^* q_f \langle y \rangle$
  where $\langle x \rangle$ and $\langle y \rangle$ are encodings of the integers x,y
- We need to encode integers as strings over some alphabet
  - Eg. Weighted positional using decimal representation etc.
- Question: what is the 'simplest' representation ?

9

Turing Machine as a computer of functions: Encoding integers

- The simplest scheme to encode an integer uses the Unary representation
  - One symbol a, and integer $n \geq 0$ is represented as a^n
  - Unary representation is what we start with….
- Our notation: We will use symbol 0 to as the unary symbol
  - Integer $n > 0$ encoded as 0^n
  - Note: could pick any other symbol…1, 2, ….
    - Textbook uses 1 instead of 0
- Given an integer n, a TM computers $f(n)$ if

$$q_0\, 0^n \vdash_M^* q_f\, 0^{f(n)} \text{ where } q_f \text{ is a final state.}$$

  - *Ex: $q_0\, 0^n \vdash_M^* q_f\, 0^{2n}$ computes $f(x) = 2x$*

10

Turing Machine as a computer of functions: Encoding integers

- What if we have multiple arguments…each encoded in unary ?
  - Need another symbol to separate the arguments
- To specify two arguments x,y ?
- Given inputs x,y a TM M computers $f(x,y)$ if:
- *Generalize to any n arguments:*
- What about multiple outputs ?

11

Turing Machine as a computer of functions: Encoding integers

- What if we have multiple arguments…each encoded in unary ?
  - Need another symbol to separate the arguments
- Notation: we use 1 to separate arguments
  - Textbook uses 0 (it swaps the roles of the 0's and 1's)
- To specify two arguments x,y we place 0^x10^y on the tape.
- Given inputs x,y a TM M computers $f(x,y)$ if

$$q_0\, 0^x10^y \vdash_M^* q_f\, 0^{f(x,y)} \text{ where } q_f \text{ is a final state.}$$

  - Ex: $q_0\, 0^x10^y \vdash_M^* q_f\, 0^{x+y}$ computes $f(x,y) = x+y$
- *Generalize to any n arguments:* $q_0\, 0^{x1}10^{x2}1..10^{xn} \vdash_M^* q_f\, 0^{f(x1,x2,\ldots xn)}$
- What about multiple outputs ? Again, place separators (use different separator such as 11 to distinguish?)

$$q_0\, 0^x10^y \vdash_M^* q_f\, 0^{f(x,y)}110^{g(x,y)}$$

12

Definition: Functions

- A function $f(x_1,x_2,\ldots,x_n)$ computed by a Turing machine is called a *partial recursive function*
  - Analogous to Recursively Enumerable (r.e.) languages
    - TM may not halt on all inputs
- If a function $f(x_1,x_2,\ldots,x_n)$ is defined for all values of $x_1,x_2,\ldots,x_n$ and computed by a Turing machine, then we say that f is a *total recursive function*
  - Analogous to recursive languages
  - TM halts on all inputs since function value is defined for all inputs
- The terms recursive and partial recursive functions were introduced independent of TMs.

13

Example 1: $f(x,y) = x+y$

- Representation Model: (a) blanks are ignored or (b) no blanks in the middle of an output value
  - We will use the "no blank tape symbols in the middle of an output value"
- $f(x,y) = x+y$
  - $0^x10^y \vdash^* 0^{x+y}$
  - Algorithm: Copy every 0 to the left of 1 to the right end.
    - Copy 0^x to right after 0^y
    - Erase the 1, and halt
  - $0^x10^y \vdash^* 0^{x-i}\,10^{y+i} \vdash^* 10^{x+y} \vdash^* 0^{x+y}$

14

Example 1: *f(x,y) = x+y*

- Algorithm
- $0^x10^y \vdash^* 0^{x-i}\,10^{y+i} \vdash^* 10^{x+y} \vdash^* 0^{x+y}$

1. Read 0, change to B, move right (to copy the 0)

   IF you read 1 write B and goto 4

   - If you read a 1 then it means you are done copying the n 0's
1. Skip right over 0's and 1 until the first B: change B to 0, move left
2. Skip left over 0/1 until you read a B then go right and goto 1.
   - i.e., searching for the rightmost 0 in 0^n
3. Accept

15

Exercise 1: *f(x,y) = x -y*

- $f(x,y) = x\text{-}y$ assume for simplicity that $x \geq y$
- Algorithm: Input is 0^x10^y (tape head is at left end)
- $q_0\, 0^x10^y \vdash^* q_f\, 0^{x-y}$

16

Example 2: *f(x,y) = x*y (Multiplication)*

- How do we write a program to multiply two integers, assuming we have a program to ADD two integers ?
  - Multiplication through repeated addition

```
i= x
while i > 0
      i= i-1
      mult = ADD(mult,y)    /* subroutine call !! */
```

17

Example 2: *f(x,y) = x*y* (Multiplication)

- $f(x,y) = x*y$ assume for simplicity that $x,y \geq 1$
- $q_0 0^x 1 0^y \vdash^* q_f 0^{x*y}$
- We have TMs (programs) to (1) ADD and (2) SUBTRACT
- How do we write a program to multiply two integers, assuming we have a program to ADD two integers ?
  - Multiplication through repeated addition

18

Subroutines in TMs

- A TM has a start state and a final state
  - $p_0\, w \vdash^* p_f\, f(w)$
- How do we "call" the subroutine ?
- How do we "return" from subroutine ?
- Anything else we need to do before calling (or after returning)?

19

Example 2: *f(x,y) = x*y* (Multiplication):

Notations and Specifications

- Input is 0^x10^y and output should be 0^{x*y}
- Unlike addition and subtraction, we need to keep 0^{x-i} on the tape to keep track of value of i, and we need to keep 0^y on tape so we can keep "adding" to the running sum Mult
- Therefore introduce new output field – how ?

20

Example 3: $f(x,y) = x*y$ (Multiplication):
Notations and Specifications

- Input is 0^x10^y and output should be 0^{x*y}
- $q_00^x10^y \vdash^* \; 0^x10^y1 \;\; \vdash^* 0^{x-i}10^y1\; 0^{i*y} \vdash^* 10^y1\; 0^{x*y} \;\; \vdash^* q_f0^{x*y}$

21

Example 2: $f(x,y) = x*y$ (Multiplication):
specifications of the ADD subroutine

- Key to the Multiplication is repeated calls to the "ADD"
  - First step is specification of this "subroutine"
- This subroutine essentially copies n 0's from left of 1 to the right of 1
- Specifications: $q_10^n10^k \vdash_{ADD}{}^* q_n0^y10^{k+n}$
- Algorithm: for every 0 to the left of the 1, copy to the right end (write 0);

  repeat until no more 0's to the left
  - Similar to the ADD TM that we designed but…..

22

Example 2: *f(x,y) = x*y* (Multiplication): specifications of the ADD subroutine

- Important: we want to call this ADD subroutine (to add *y* to running sum) repeatedly…so don't "lose" (erase) the 0^n

  => at the end, after copying the n 0's we should be able to restore 0^n

  => instead of erasing the 0, write a new symbol 2 and then when done restore 2's to 0's

23

Example 2: *f(x,y) = x*y* (Multiplication): specifications of the ADD subroutine

- $q_1 0^n 1 0^k \vdash_{ADD}^* q_n 0^y 1 0^{k+n}$

24

Example 2: *f(x,y) = x*y* (Multiplication): specifications of the ADD subroutine

1. Read 0, change to 2, move right (to copy the 0) IF you read 1 then goto 4
   - If you read a 1 then it means you are done copying the n 0's
2. Skip right over 0's and 1 until the first B: change B to 0, move left
3. Skip left over 0/1 until you read a 2 then go right and goto 1.
   - i.e., searching for the rightmost "checked" 0 in 0^n
4. (Restore 0^n) Change all 2's to 0's – keep going left changing 2's to 0's until you see a 1 (the left of 1 is another argument 0^x)

25

Example 2: *f(x,y) = x*y* (Multiplication)

- We have subroutine for ADD…we need to call it repeatedly

```
i= x
while i > 0
        i= i-1
        mult = ADD(mult,y) /* subroutine call !! */
```

$$q_0 0^x 1 0^y \vdash^* \; 0^x 1 q_1 0^y 1 \vdash_{ADD}{}^* \; 0^{x-i} 1 q_5 0^y 1 \; 0^{i*y} \vdash^* \; q_f 0^{x*y}$$

To call ADD, go to state q_1

When ADD returns it is in state q_5

26

Example 2: *f(x,y) = x*y* (Multiplication)

- Calling and returning from Subroutine

27

Example 2: *f(x,y) = x*y* (Multiplication)

$q_0 0^x 1 0^y \vdash^* 0^x 1 q_1 0^y 1 \vdash_{ADD}^* 0^{x-i} 1 q_5 0^y 1\ 0^{i*y} \vdash^* q_f 0^{x*y}$

1. First (erase 0, ie., x-1) and go all the way to the end of input and add 1 and move left. /*this creates space for the output */
2. Go left to a 1, move right and call ADD
3. (return from ADD) Go left (to end of 0^{x-i})
   - Skip 0/1 until read B move right
4. Read 0, Erase 0, move right;

   IF read 1 (then i=0) write B and goto 6.
1. Skip over 0's until read 1, move right and Call ADD (this returns in step 3)
2. (Done with Mult) so erase 10^y1
   - Read 0, Write B, go right until Read 1, Write B and accept.

28

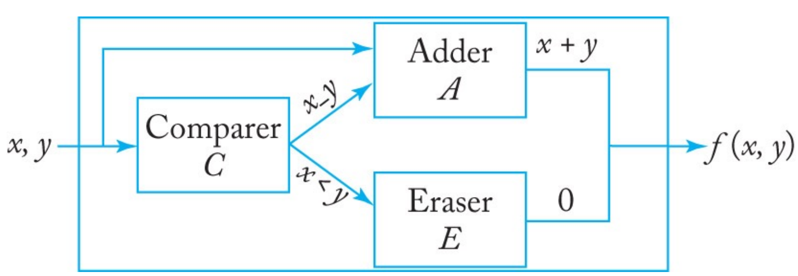
Questions/Exercises

- 1: f(n) = n^2 $q_0\, 0^n \vdash^* q_f\, 0^y$ where $y=n^2$
  - Think of what ”pre-processing” needs to be done before calling the multiplication “function”
- 2: f(n) = 2^n
  - Input is 0^n and output should be 0^y where $y=2^n$

29

Taking stock: Combining Turing Machines

- By combining Turing Machines that perform simple tasks, complex algorithms can be implemented
- Example: assume the existence of
  - a machine to compare two numbers (comparer)
  - Machine to add two numbers (adder)
  - machine to erase the input (eraser)
- TM to compute function f(x, y) = x + y (if x ≥ y), 0 (if x < y)



30

Next…Modifications to standard Turing machine

- Add new features/capabilities to standard turing machine….does this increase power/capabilities ?
  - Tape with multiple tracks
  - Tape with “stay” option
  - Semi-infinite tape
  - Multiple tapes
  - Multidimensional tapes
  - Non-deterministic Turing machines
- Turns out they are all equivalent to the standard TM
- Proofs: simulation of these models on the standard TM

31