Theory of Computation

Lesson 16a - Turing Machines Intro continued

Review

Last Time:

1. Proof of Pumping Lemma for CFLs.
   - If in the parse tree of w some path from root to leaves includes a repeated variable, there corresponds a partition $w = uvxyz$ with pumpable v, y. Some arrangement then guarantees $v \neq \varepsilon$ or $y \neq \varepsilon$.
   - If $|w| \geq p$ then every parse tree of w is "tall" and includes a root-to-leaf path with repeated variables.
2. Turing Machines Intro
   - Definition
   - Transitions are quintuples $(q, a)\ (p, b, M)$
   - Left moves ignored at left end of tape
   - Configurations

Configurations

Poll 16.1 Applied to configuration

$\sqcup\ a\ \sqcup\ a\ q\ b\ a\ b$

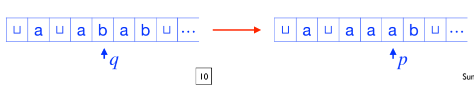
transition $(q, b)\ (p, a, R)$ yields

(a) $\sqcup\ a\ \sqcup\ a\ p\ b\ a\ b$

(b) $\sqcup\ a\ \sqcup\ p\ a\ b\ a\ b$

(c) $\sqcup\ a\ \sqcup\ p\ a\ a\ a\ b$

(d) $\sqcup\ a\ \sqcup\ a\ a\ p\ a\ b$ ← correct



Turing Machines

Formal Definition. A (deterministic) Turing Machine is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, s, q_{accept}, q_{reject})$$

where

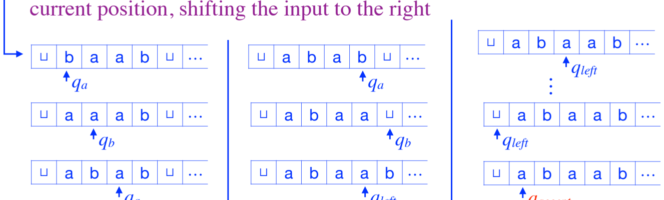
1. Q is a finite set whose elements are called states.
2. Σ is the input alphabet not containing the blank symbol $\sqcup$.
3. Γ is the tape alphabet, with $\Sigma \subseteq \Gamma$ and $\sqcup \in \Gamma$
4. $\delta: (Q - \{q_{accept}, q_{reject}\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function.
5. $s \in Q$ is the start state.
6. $q_{accept} \in Q$ is the accepting state.
7. $q_{reject} \in Q$ is the rejecting state with $q_{accept} \neq q_{reject}$.

Turing Machines

Exercise. Design a Turing Machine that shifts its input one cell to the right (and finishes in state q_{accept} under the second cell).

$Q = \{s, q_{accept}, q_{reject}\} \cup \{q_{left}\} \cup \{q_x \mid x \in \Sigma\}$

For every $x \in \Sigma$ there is a state q_x whose function is to insert x at the current position, shifting the input to the right



Turing Machines

Exercise. Design a Turing Machine that shifts its input one cell to the right (and finishes in state q_{accept} under the second cell).

$Q = \{s, q_{accept}, q_{reject}\} \cup \{q_{left}\} \cup \{q_x \mid x \in \Sigma\}$

For every $x \in \Sigma$ there is a state q_x whose function is to insert x at the current position, shifting the input to the right

$(s\ \sqcup)\ (q_{accept}\ \sqcup\ R)$ // handle empty input

$(s\ x)\ (q_x\ \sqcup\ R)$ // for every $x \in \Sigma$

$(q_x\ y)\ (q_y\ x\ R)$ // for all $x, y \in \Sigma$

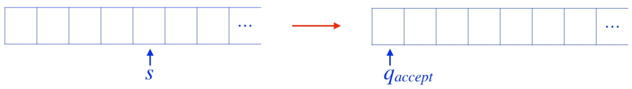
$(q_x\ \sqcup)\ (q_{left}\ x\ L)$ // for every $x \in \Sigma$

$(q_{left}\ x)\ (q_{left}\ x\ L)$ // for every $x \in \Sigma$

$(q_{left}\ \sqcup)\ (q_{accept}\ \sqcup\ R)$

Turing Machines

Exercise. Design a Turing Machine that, started with the head anywhere on a blank tape, will finish with a blank tape and the head on the first cell in state q_{accept}.



$\Gamma = \Sigma \cup \{\sqcup, \$\}$ $\quad Q = \{s, q_{accept}, q_{reject}\} \cup \{q_{erase}, q_{left}\}$

(1) Locate first cell

$(s\ \sqcup)\ (s\ \$\ L)$

$(s\ \$)\ (q_{erase}\ \$\ R)$

(2) Move right erasing

$(q_{erase}\ \$)\ (q_{erase} \sqcup R)$

$(q_{erase}\ \sqcup)\ (q_{left} \sqcup L)$

(3) Return left

$(q_{left}\ \sqcup)\ (q_{left} \sqcup L)$

$(q_{left}\ \$)\ (q_{accept} \sqcup L)$

Theory of Computation

Lesson 16b - Decidable vs. Recognizable Languages

Deciders vs Recognizers

Given $L \subseteq \Sigma^*$, deterministic Turing machine M with input alphabet Σ.

Definition. M decides L if for every $w \in \Sigma^*$

$w \in L \Rightarrow M$ accepts w

$w \notin L \Rightarrow M$ rejects w

} M halts on every input w

Definition. M recognizes (accepts) L if for every $w \in \Sigma^*$

$w \in L \Rightarrow M$ accepts w

$w \notin L \Rightarrow M$ does not accept w ⟵ M rejects or loops

In both cases $L = L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$

Deciders vs Recognizers

M decides L if for every $w \in \Sigma^*$	M recognizes L if for every $w \in \Sigma^*$
$w \in L \Rightarrow M$ accepts w	$w \in L \Rightarrow M$ accepts w
$w \notin L \Rightarrow M$ rejects w	$w \notin L \Rightarrow M$ does not accept w

Note: M decides $L \Rightarrow M$ recognizes L

later: exist languages that can be recognized but that can't be decided.

Run M on input w	if $w \in L$ will we find out?	if $w \notin L$ will we find out?
M decides L	yes	yes
M recognizes L	yes	maybe not

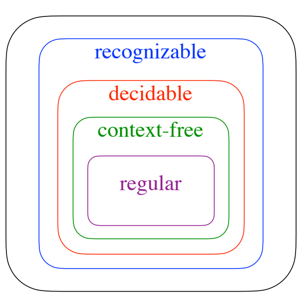
Deciders vs Recognizers

L is decidable: there is a Turing machine that decides L.

computable, recursive

L is recognizable: there is a Turing machine that recognizes L.

computably enumerable, recursively enumerable



all languages $L \subseteq \Sigma^*$

Still to show:
recognizable $\neq$ all
decidable $\neq$ recognizable
context-free $\subsetneq$ decidable

Decider example: even length words

Example: Design a Turing machine that decides the language $L = \{w \in \{a, b\}^* \mid |w| \text{ is even}\}$

Solution. L is regular. Simulate a DFA for L.

$M = (Q, \Sigma, \Gamma, \delta, s, q_{accept}, q_{reject})$

$\Sigma = \{a, b\}$

$\Gamma = \Sigma \cup \{\sqcup\}$

$Q = \{q_{even}, q_{odd}, q_{accept}, q_{reject}\}$

$s = q_{even}$

Transitions δ:

q_{even}	a	q_{odd}	a	R
q_{even}	b	q_{odd}	b	R
q_{odd}	a	q_{even}	a	R
q_{odd}	b	q_{even}	b	R
q_{even}	$\sqcup$	q_{accept}	$\sqcup$	R
q_{odd}	$\sqcup$	q_{reject}	$\sqcup$	R

Decider example: even length words

Tabular representation.

δ	a	b	$\sqcup$
$\rightarrow q_{even}$	$q_{odd}\ a\ R$	$q_{odd}\ b\ R$	$q_{accept}\ \sqcup\ R$
q_{odd}	$q_{even}\ a\ R$	$q_{even}\ b\ R$	$q_{reject}\ \sqcup\ R$

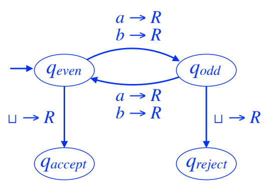
Diagram representation.

For each $(q,\ a)\ (p,\ b,\ \text{move})$

$q \xrightarrow{a\ \to\ b,\ \text{move}} p$

If $b = a$, just write

$q \xrightarrow{a\ \to\ \text{move}} p$



Decider example: more a's than b's

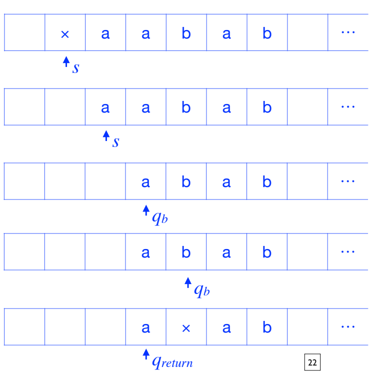
Exercise. Design a Turing machine that decides the language $\{w \in \{a, b\}^* \mid w$ has more a's than b's$\}$

Generalize. Given piece of text consisting of a's, b's and $\times$'s, decide whether there are more a's than b's.

Solution. Proceed in passes. Each pass removes one a and one b until only a's, or only b's, or none remain.

Decider example: more a's than b's

Exercise. Design a Turing machine that decides the language $\{w \in \{a, b\}^* \mid w$ has more a's than b's$\}$



Solution. Proceed in passes. Each pass removes one a and one b until only a's, or only b's, or none remain.

Decider example: more a's than b's

Exercise. Design a Turing machine that decides the language $\{w \in \{a, b\}^* \mid w$ has more a's than b's$\}$

Solution. Proceed in passes. Each pass removes one a and one b until only a's, or only b's, or none remain.

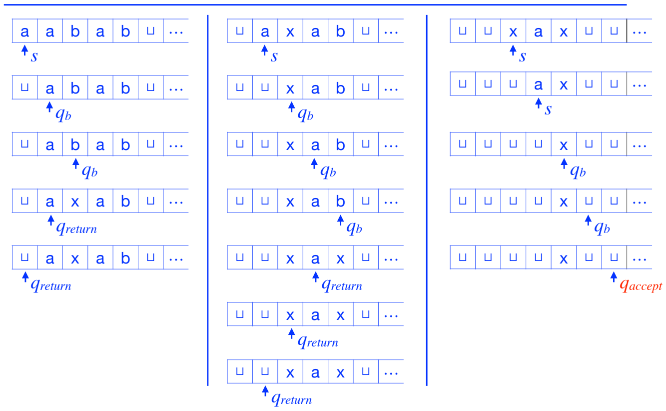
Pass begins at left end of text in state s looking for symbol of $\{a, b\}$

- If a found, erase, switch to state q_b looking for b.
- If b found, erase, switch to state q_a looking for a.
- Once q_a , q_b find target a or b, replace with $\times$ and return left.

Invariants: (1) text to process never includes a blank.
(2) when in state s, tape to the left is blank, text to process begins at head position.

Decider example: more a's than b's

Exercise. Decider for $\{w \in \{a, b\}^* \mid w$ has more a's than b's$\}$



Decider example: more a's than b's

$M = (Q, \Sigma, \Gamma, \delta, s, q_{accept}, q_{reject})$ $\quad \Sigma = \{a, b\}$

$Q = \{s, q_{accept}, q_{reject}\} \cup \{q_a, q_b, q_{return}\}$ $\quad \Gamma = \Sigma \cup \{\sqcup\} \cup \{\times\}$

Transitions δ:

s	$\sqcup$	q_{reject}	$\sqcup$	R	q_{return}	$\sqcup$	s	$\sqcup$	R
s	a	q_b	$\sqcup$	R	q_{return}	a	q_{return}	a	L
s	b	q_a	$\sqcup$	R	q_{return}	b	q_{return}	b	L
s	$\times$	s	$\sqcup$	R	q_{return}	$\times$	q_{return}	$\times$	L
q_a	$\sqcup$	q_{reject}	$\sqcup$	R	q_b	$\sqcup$	q_{accept}	$\sqcup$	R
q_a	a	q_{return}	$\times$	L	q_b	a	q_b	a	R
q_a	b	q_a	b	R	q_b	b	q_{return}	$\times$	L
q_a	$\times$	q_a	$\times$	R	q_b	$\times$	q_b	$\times$	R

Decider example: more a's than b's

```
# decides whether strings over
# {a,b} have more a's than b's

start: start
accept: good
reject: bad

transitions:
- [start, _, bad, _, R]
- [start, a, findb, _, R]
- [start, b, finda, _, R]
- [start, x, start, _, R]

- [finda, _, bad, _, R]
- [finda, a, return, x, L]
- [finda, b, finda, b, R]
- [finda, x, finda, x, R]

- [findb, _, good, _, R]
- [findb, a, findb, a, R]
- [findb, b, return, x, L]
- [findb, x, findb, x, R]

- [return, _, start, _, R]
- [return, a, return, a, L]
- [return, b, return, b, L]
- [return, x, return, x, L]
```

tint note:
- template slightly different from dfa's: need to specify the names of start, accept, and reject (can be any names),
- transitions are quintuples,
- tint uses the underscore _ as blank symbol.

to run (with test + verbose):

```
./tint -m one-way-tm -t -v more_as.txt "a a b a b"
```

can also use
https://tintgenerator.vercel.app
(has fixed names for accept and reject)

Decider example: more a's than b's

```
./tint  -m  one-way-tm  -t  -v  more_as.txt  "a a b a b"

start: a a b a b _
      ^
findb: _ a b a b _
        ^
findb: _ a b a b _
          ^
return: _ a x a b _
         ^
return: _ a x a b _
       ^
start: _ a x a b _
        ^
findb: _ _ x a b _
          ^
findb: _ _ x a b _
            ^
findb: _ _ x a b _
              ^
```

```
return: _ _ x a x _
             ^
return: _ _ x a x _
           ^
return: _ _ x a x _
         ^
start: _ _ x a x _
          ^
start: _ _ _ a x _
            ^
findb: _ _ _ _ x _
              ^
findb: _ _ _ _ x _
                ^
accept: _ _ _ _ x _ _
                   ^
Accepted.
```


Theory of Computation

Lesson 16d - More Examples

Example: deciding $w \# w$

Exercise. Design a Turing machine with $\Sigma = \{a, b, \#\}$ that decides $\{w \# w \mid w \in \{a, b\}^* \}$ ← not context-free, only $w \# w^R$ context-free

Idea. Given input with exactly one # (reject otherwise)
input $= w_1 \# w_2$ ← with $w_1 \in \{a, b\}^*$, $w_2 \in \{a, b\}^*$

repeat
1. mark first unmarked symbol of w_1, remember symbol in state q_a or q_b.
2. go right of #, transition to q_a' or q_b', find first unmarked symbol, compare with symbol remembered in state. If different reject. If equal, mark.

if in (1) no unmarked symbol of w_1 found, check that all symbols of w_2 are marked; accept if yes, reject if no.

if in (2) no unmarked symbol of w_2 found, reject.

a	b	b	#	a	b	a		…
×	b	b	#	a	b	a		…
×	b	b	#	×	b	a		…
⋮								
×	×	b	#	×	×	a		…
×	×	×	#	×	×	a		…

↑ q_b' ↑ q_{reject}