Homework 8

Monday, November 11, 2024 12:17 PM

1. 3.1 a) 0

$q_1 0$
$\sqcup q_2$
$\sqcup q_{accept}$

3.1 c) 000

$q_1 000$
$\sqcup q_2 00$
$\sqcup X q_3 0$
$\sqcup X 0 q_4$
$\sqcup X 0 q_{reject}$

2. 3.2b) 1#1

$q_1 1\#1$
$X q_3 \#1$
$X \# q_5 1$
$X q_6 \# X$
$q_7 X \# X$
$X q_1 \# X$
$X \# q_8 X$
$X \# X q_8$
$X \# X q_{accept}$

3.2 c) 1##1

$q_1 1\#\#1$
$X q_3 \#\#1$
$X \# q_5 \#1$
$X \# q_{reject} \#1$

3. 1. Loop through input, mark an a w/ X or a c w/ Y and look for the corresponding input and mark that.
   - Either a and c or c and a
   2. If cannot find a corresponding a or c, reject.
   3. Continue until no a and c in input.

4. Loop through input, marking X w/ Z and b w/ W.
5. If there is an X w/ no corresponding b; reject. (a > b)
6. If there is a b w/ no X, accept. (a < b)
7. If neither is found (a = b), reject.

4. To mimic the function of a Turing machine w/ a 2-PDA, first we loop through input into the first stack. Thus, the input is accessible in backwards order. Then, we push and pop the entire first stack into the second, and now we have the input in order. This position indicates the start state. We simulate Turing transitions by using the first stack as the left of the state and the second stack as the right of the state.

- For a left transition...

$(q_i, x) \to (q_j, A, L) \Rightarrow (q_i, X, \varepsilon) \to (q_j, A, \varepsilon)$

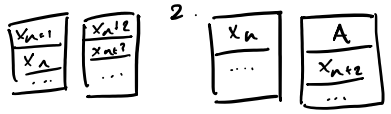
(pop: 1 2; push: 1 2)

In a turing machine...

1. $\ldots X_n X_{n+1} q_i X_{n+2} \ldots$
2. $\ldots X_n q_j A X_{n+2} \ldots$

$\approx$

In a 2-PDA...



Thus, the 2-PDA maintains the "left" and "right" sides of the state.

- For a right transition...

$(q_i, x) \to (q_j, A, R) \Rightarrow (q_i, \varepsilon, x) \to (q_j, \varepsilon, A)$

5. 3.8 c)
1. Loop over input, cross out one 1 and 2 0s
2. If both conditions are not met, accept
   - i.e. One 1 and zero 0s
   - Implies $2101 \neq 111$
3. If there is no remaining 0s AND 1s, reject
4. Repeat at 1

Thus, if removing twice as many 0s as 1s removes the entire string, then $2101 = 111$, and we reject.
Any other configuration accepts once input is over, thus this machine halts on all inputs $\Rightarrow$ decidable
$\Rightarrow$ decides $\{w \mid w$ does not contain twice as many 0s as 1s$\}$

6. Let M_1 recognize L_1 and M_2 recognize L_2, s.t. L_1 & L_2 are Turing recognizable.

Let machine M accept $L = L_1 \cup L_2$

- w runs both M_1 and M_2 on an input.
- If either accepts, they accept
- Thus, $L(M) = L_1 \cup L_2$

$w \in L \Rightarrow w \in L_1$ or $w \in L_2$

$w \in L_1$ or $w \in L_2 \Rightarrow w \in L_1 \cup L_2$

Thus, Turing recognizable languages are closed under union.

7. Let L_1 and L_2 be Turing decidable languages, and M_1 and M_2 that decide them.

Let machine M accept $L = L_1 L_2$

- Let w first get read by M_1
- M_1's accept state leads to M_2's start
- w read by M_2, and M accepts if M_2 accepts

Thus, M accepts strings $L = L_1 L_2$

M_1 decides $\Rightarrow$ M_1 halts

M_2 decides $\Rightarrow$ M_2 halts

M runs M_1 then M_2, thus,

M halts $\Rightarrow$ M decidable

$\Rightarrow$ decidable languages closed under concat...

8. Let L_1 and L_2 be Turing decidable languages, and M_1 and M_2 that decide them.

Let M accept $L_1 \cap L_2$ such that...

- Input w is run on both M_1 and M_2
- Only accepts if both M_1 and M_2 accept

Thus, M accepts $L_1 \cap L_2$

M_1, M_2 decidable $\Rightarrow$ M_1, M_2 halt

$\Rightarrow$ M halts

M halts and decides $L = L_1 \cap L_2 \Rightarrow M$ decidable

$\Rightarrow$ Turing decidable closed under intersection

9. Let M_1 recognize L_1 s.t. L_1 is Turing recognizable.
Let M be nondeterministic and accept L_1^*.
- Start state can lead to accept state
- accept state can lead back to start state

Thus, M can accept any combination of $w \in L_1$

10. 3.16) Enumerator enumerates language in standard string order $\Leftrightarrow$ decidable
- Enumerates $\Rightarrow$ decidable
  E enumerates L in order.
  We can create turing machine M that accepts all strings that E enumerates and rejects those not enumerated by E
- Decidable $\Rightarrow$ Enumerates
  L is decidable $\Rightarrow \exists M$ that halts on all $w \in L$

  Thus, we can create an enumerator to generate all possible strings in order and only enumerate if accepted. This is possible because decidable $\Rightarrow$ every string halts, so we can generate w/o infinite loops.

Thus, decidable $\Leftrightarrow$ enumerable in standard string order