CS452 HW6

Yuting Chen

3.9

(1) The tasks performed by 1-PDAs can be done by 2-PDAs.

The language $L=\{w \mid w \in a^n b^n c^n\}$ that accepted by 2-PDAs but not by 1-PDA. So it's not CFL.

If there exist 2 stacks. As follow: Read $a^* b^* c^*$

- PDA is in state q_{accept}. Reading "a" push it in stack 1. goto q_1. Otherwise, reading a "b" or "c" go to q_{reject}. The remaining input stay here.
- In state q_1. Reading input "a", push it into stack 1, and go to q_2 in case. then "b" is read and pop one "a" from stack 1 and push "b" on stack 2. If "c" is read then go to q_{reject}. The remaining input stay here.
- In state q_2. Reading input "b", pop "a" and push "b" in stack 2. and go to q_3. then "c" is read and pop "b" from stack 2. If there is no "a" on stack with "b" in hand, then go to q_{reject}. "a" is read again.
- In state q_3, Reading input "c", pop "b" from stack 2. Reject in case "a" or "b" is read.
- If both of stacks are empty, accept.

So, L is recognized by 2-PDA.

Thus, 2PDAs are more powerful then 1-PDAs.

(2) The 3-PDAs also can do the tasks that is done by 2-PDAs.

- Insert a input tape in stack 2. Its left end is on top. Entire string can be put inside the stack from stack 1. The input is read from left, it's left most will be at bottom of stack. On popping from stack 1 and then pushing the popped symbol into stack 2, the result is retrieved.
- The symbol that Turing machine reads is at top of stack 2 and its current symbol.
- If "a" is the current symbol that is to be read is "a". So Turing Machine moves left, write "b" then "a" can be popped from stack 2 and "b" can be pushed into stack. After a symbol can be popped from stack 1 to push it to stack 2.
- If Turing machine have made a correct movement. "a" will have popped from stack 2 and symbol "b" is pushed in stack 1.

So, it proves the TM are powerful as 3-PDA.

Thus, it can be concluded that 3-PDAs are no powerful than 2-PDAs. Both PDAs are equally powerful.

3.15

b. concatenation.

Let L_1 and L_2 to be two decidable Language.

Let M_1 and M_2 to be the Turing machines that decides L_1 and L_2 respectively.

We construct a Tm M' that recognizes $L_1 \circ L_2$: On input w.

$\Rightarrow L(M') = L_1 \circ L_2$

- Split w into two parts w_1, w_2. $w = w_1 w_2$
- Run M_1 on w_1. If it's rejected then Output reject.
- Else run M_2 on w_2. If it's rejected then Output reject.
- Else output accept.

If the first part is accepted by M_1, the second part is accepted by M_2, then w is accepted by M'.

Else w doesn't belong to the concatenation and it's rejected.

Thus, $L(M') = L_1 \circ L_2$. The Turing-recognizable languages are closed under concatenation.

c. Star

Let L be a Turing decidable Language.

Let M be the Turing machine that decides L.

We construct a Tm M' that recognizes L^*: On input w.

$L(M') = L^*$

- Split w into n parts. $w = w_1 w_2 w_3 \cdots w_n$
- Run M on w in different ways. Let $w = w_i$ for $i = 1, 2, 3 \cdots n$.
- If M accepts each of strings w_i, then output accept.
- If we tried all of the cuts, and all of cuts are not success, then output reject.

When w is cut into every string that is accepted by M, then w belongs to the star of L.

So, M' accepts w in every different ways. Else w will be rejected. w_i for $i = 1, 2, 3 \cdots n$.

Thus, $L(M') = L^*$. The Turing-recognizable languages are closed under star operation.

d. complementation.

Let L be a Turing decidable Language.

Let M be the Turing machine that decides L.

We construct a complement is M': On input w

- If M reject, then output accept.
- Else reject.

M' does the opposite of M. what ever M is.

Thus. The Turing-recognizable languages are closed under complementation.

(a) Intersection

Let L_1 and L_2 to be two decidable Language.

Let M_1 and M_2 to be the Turing machines that decides L_1 and L_2 respectively.

We construct a Tm M' that recognizes $L_1 \cap L_2$: On input w.

$\Rightarrow L(M') = L_1 \cap L_2$

- Run M_1 on w. If M_1 rejects then output reject.
- Run M_2 on w. If M_2 rejects then output reject.
- Else accept.

If either of M_1 and M_2 rejects then M' rejects w.

If both of M_1 and M_2 accepts then M' accepts w.

Thus, $L(M') = L_1 \cap L_2$. The Turing-recognizable languages are closed under intersection.

3.18

Assume that we have a Turing Machine M to decide a language L.

Then, we use M to construct an enumerator E.

Generate the strings in Lexicographic order.

Then, input each of strings into M for language L.

If M accepts then print that string.

So, E prints all strings of language L in Lexicograph order.

After, we consider the other direction.

If we have an E for L, then we could use E to construct a M that decides L.
Case 1: L is finite, then it is decidable, since all finite languages are decidable.
Case 2: L is infinite.
Input w, the decidor enumerates all strings of L in Lexicographic order until a string appears. It greater than w in Lexicographic order. It must occur. because we know that L is infinite.
If w appeared in the enumeration. then accept.
If w didn't appear in the enumeration, it will never appear, So output reject.
As above 2 cases, L is both decidable.
Thus, A language is decidable iff some enumerator enumerates the language in lexicographic order that is true.

3.19

Let L be an infinite Turing recognizable language.
Let M be the enumerator that enumerates L.
Let $L' = w_1, w_2, w_3 \cdots$ $L' = w_i$ for $i > 1$
Assume L' is finite.
w_i be the last string enumerated by M.
Then, lexicographically largest element in L'.
All strings enumerated by M must lexicographically less than w_i.
$\Rightarrow$ There are only a finite number of string that are less than w_i.
So, That is a contradiction. L' is infinite.
All strings in L' were at some point enumerated by M.
So, L' is subset of L.
After we will prove that L' is decide for given enumerator M, we construct M' in lexicographic order.
Let w be the last string that M' emitted. Initialize w to a dummy value that comes lexicographically before all strings.
Simulate M until it emits string s.
If $s > w$ lexicographic, then set $w = s$, M' emits s.
Else, we can dismiss s. and go back simulating M.

So, we get that M' is constructed and it enumerated L' in lexicographic order.
From Question 3.18. We know that.
A language is decidable iff some enumerator enumerates the language in lexicographic order.
So, L' is decidable.
In sum, every infinite Turing-recognizable language has an infinite decidable subset.

4.11 Let $\text{INFINITE}_{\text{PDA}} = \{\langle M \rangle \mid M \text{ is a PDA and } L(M) \text{ is an infinite language}\}$. Show that $\text{INFINITE}_{\text{PDA}}$ is decidable.

Bulid a Turing Machine that will do as follow:
- Reading M and creating an equivalent context-free grammar G.
- Convert G to CNF, name it G'.
- Generate a graph A
- Add edges (T, U) from T to U and (T, V) from T to V.
- Apply DFS or BFS from start state to see if A has a directed cycle.
- If so, accept. If not, reject.
- If $\langle M \rangle \in \text{INFINITE}_{\text{PDA}}$, then the length of string will greater than pumping length of $L(M)$.
- Using pumping lemma to prove, there is a variable V can derive sVt for strings s, t.
- The graph A has a cycle involving variable V.
- Assume there is a cycle in A.
- $V \rightarrow^* sVt$ must exist, s or t must be a non-empty.
- Since there is a scope for finding a cycle from S. $S \rightarrow^* aVb$
- Thus, $S \rightarrow^* a^i V b^i$. $L(M)$ is infinite.

Thus, $\text{Infinite}_{\text{PDA}}$ is decidable.

4.13 Let $A = \{\langle R, S \rangle \mid R \text{ and } S \text{ are regular expressions and } L(R) \subseteq L(S)\}$. Show that A is decidable.

First, we need to note that E_{DFA} is decidable.

$L(R) \subseteq L(S)$ iff $\forall w \in L(R), w \in L(S)$

iff $\forall w \in L(R), w \notin L(\overline{S})$

iff $L(R) \cap L(\overline{S}) = \phi$

$\Rightarrow L(C) = \overline{L(S)} \cap L(R)$

Bulid a Turing Machine that will do as follow:

- Check string w encodes a pair $\langle R, S \rangle$ where R and S are regular expressions.
- If not, reject w.
- The conversion of regular expressions to an equivalent NFA. R into an equivalent DFA DR. S into an equivalent DFA DS.
- Build DFA $\overline{DS}$ that accepts $L(\overline{DS})$
- Build DFA D that is intersection of DR and $\overline{DS}$.
- Run TM with input $\langle D \rangle$ to determine if $L(D)$ is empty.
- If accepts $\langle D \rangle$, then accept w. ($L(D)$ is empty)
- If rejects $\langle D \rangle$, then reject w. ($L(D)$ is not empty)

Prove that the Turing Machine decides Language A.

prove Tm halts on all inputs.

Let w be any string. Check this in finite steps.

① w doesn't codify a pair of regular expressions.

then MT stops and rejects w.

② w codifies a pair of regular expressions.

Assume w codifies a pair be $\langle R, S \rangle$.

Construct DFA as described.

Regular languages are closed under intersection and complementation.

It takes finite time.

So, MT stops on any input w in finite steps.

Thus, language A is decidable.

4.18

If D exists, then we can construct a Turing Machine M.

we search each possible string y and test if $\langle x, y \rangle \in D$.

If y exists, then accept.

A machine M will accept any string in C in finite steps.
So, C is Turing recognizable.
If C is recognized by some Turing Machine M.
$\Rightarrow D = \{\langle x, y\rangle \mid M \text{ accepts } x \text{ within } |y| \text{ steps}\}$.
$\Rightarrow$ D is decidable, $x \in C$ iff there exists y such that $\langle x, y\rangle \in D$.
Thus, $C = \{x \mid \exists y \langle x, y\rangle \in D\}$.

5.22

If $A \leq_m A_{TM}$, then A is Turing-recognizable as A_{TM} is Turing-recognizable.
In case A is Turing-recognizable, there is Turing-recognizable R that recognizes A.
Bulid a Turing Machine that will do as follow:
On input w.

- Write $\langle R, w\rangle$ on the tape and halts.
- It can be checked that $\langle R, w\rangle$ is in A_{TM} if and only if w is in A.

Thus, there is a mapping reduction of A to A_{TM}.
Thus, $A \leq_m A_{TM}$.

5.23

If $A \leq_m A_{TM}$, then A is Turing-recognizable as A_{TM} is Turing-recognizable.
In case A is Turing-recognizable.
Define a function f: $f(x) = 01$ if $x \in A$
$f(x) = 10$ if $x \notin A$.
If A is decidable.
Then A can compute this function to use decider. $x \in A$ if and only if $f(x) \in A_{TM}$.
Thus, there is a mapping reduction of A to A_{TM}. $\Rightarrow A \leq_m A_{TM}$.
If $A \leq_m A_{TM}$.
Then there exist a function f, $y \in A$ if and only if $f(y) \in A_{TM}$.
Bulid a Turing Machine that will do as follow:

On input y.

- Compute the fuction $f(y)$.
- If $f(y)$ is in form 0^*1^*, then accept.
- Else, reject.

$\Rightarrow y \in A$, $f(y)$ is of the form 0^*1^* and Turing Machine accept y.

Thus, Turing machine decides A.

We got that if A is decidable then $A \leq_m B$.

if $A \leq_m B$ then is decidable.

So, A is decidable iff $A \leq_m 0^*1^*$.