دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

نظریه زبان‌ها و ماشین‌ها

امیررضا آذری - ۹۹۱۰۱۰۸۷

پاسخ تمرین چهارم

۱. پرسش نخست

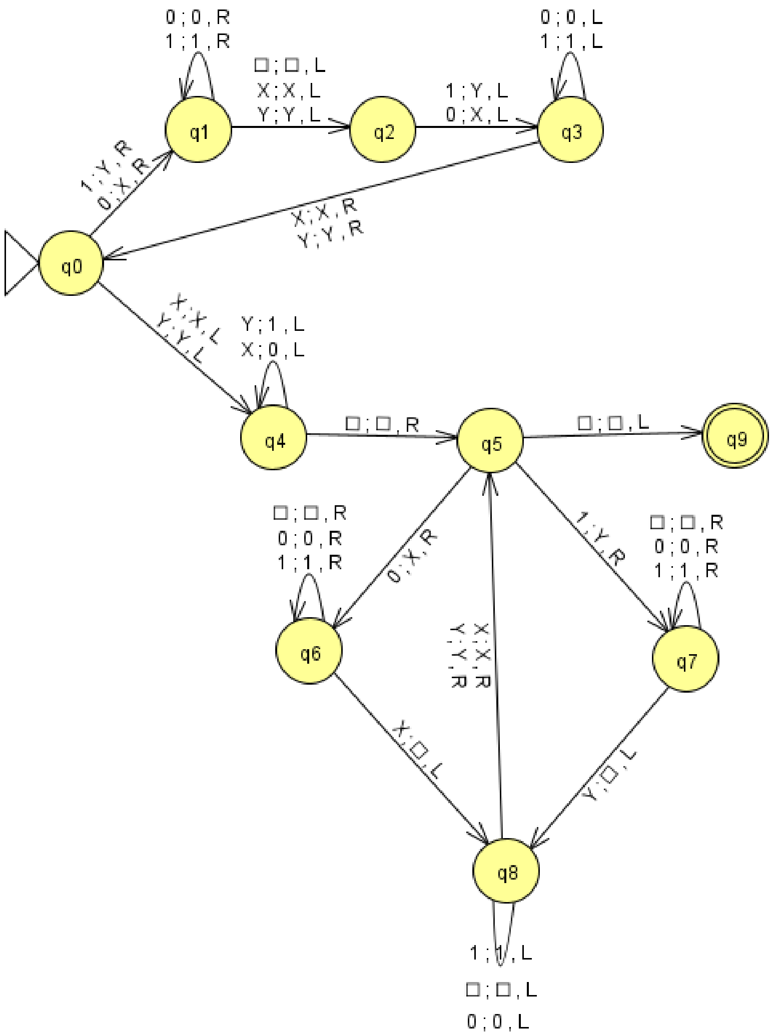
پاسخ.
برای تمام بخش‌های این سوال توصیف نموداری خواهیم آورد و همچنین توضیح نیز خواهمیم داد.

الف)

برای یافتن ماشین تورینگ این زبان، باید نقطه وسط آن را بیابیم. بنابراین در ابتدا از ابتدای رشته ۰ و ۱ ها را به x و y تبدیل می‌کنیم و به سمت مرکز خواهیم آمد. آنقدر ادامه می‌دهیم تا تنها x و y در رشته باقی مانده باشد. بعد از رسیدن به این مرحله، بخش سمت چپ را به حالت قبلی باز می‌گردانیم. حال از ابتدای ریشه شروع می‌کنیم و مجددا اگر ۰ دیدم آن را به x و ااگر ۱ دیدیم آن را به y تبدیل می‌کنیم. برای مثال در نظر بگیرید که ابتدای ریشه ۱ است و ما آن را به y تبدیل کردیم. حال آنقدر به راست حرکت می‌کنیم تا y ببینیم. بعد دیدن y، آن را با blank جایگزین می‌کنیم. سپس باز به چپ برمی‌گردیم تا اولین رقم را ببینیم و همین مراحل را مجددا تکرار می‌نماییم. حال اگر در انتها هیچ ۰ و ۱ای باقی نماند و تمام x و های‌y بخش راست به blank تبدیل شده باشند، اکسپت می‌کنیم. برای مثال رشته ۰۰۱۰۰۱ را در نظر بگیرید. استپ‌های زیر را طی خواهیم کرد:

$$
\begin{gathered}
۰۰۱۰۰۱\\
X۰۱۰۰۱\\
X۰۱۰۰Y\\
XX۱۰۰Y\\
XX۱۰XY\\
XXY۰XY\\
XXYXXY\\
XX۱XXY\\
X۰۱XXY\\
۰۰۱XXY\\
X۰۱\square XY\\
XX۱\square\square Y\\
XXY\square\square\square
\end{gathered}
$$

نمودار آن را با jflap به شکل زیر رسم نموده‌ایم. همچنین برای یک نمونه شبیه سازی انجام داده‌ایم که عکس‌های آن در فایل زیپ آپلود شده می‌باشند.



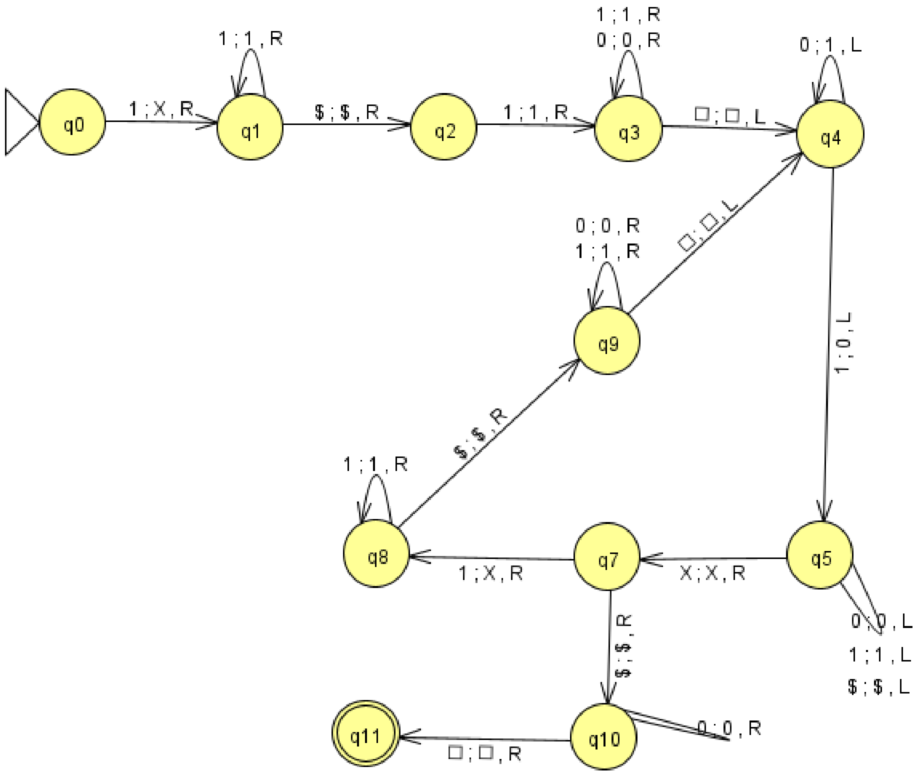
یک تست درست و یک تست نادرست نیز به این ماشین داده شده است که درستی عملکرد ماشین در تصاویر موجود در کنار این فایل، وجود دارند. همچنین برای توضیحات اضافه‌تر، این لینک کمک خواهد کرد.

ب)

ایده اجرایی این بخش این است که ما در هر مرحله، یکی از یک‌های قبل $ را مارک می‌کنیم. سپس یک واحد از عدد سمت راست کم می‌کنیم. برای کم کردم یک واحد به این شکل عمل می‌کنیم که از راست‌ترین رقم شروع کرده، تا وقتی که به اولین یک برسیم، همه صفرها را تبدیل به یک کرده و اولین یک را تبدیل به صفر می‌نماییم.
مثال:

$$۱۰۱۰۰ = ۲۰ \rightarrow ۱۰۰۱۱ = ۱۹$$

همچنین توجه کنید عدد باینری ما نباید با ۰ شروع بشود و همیشه باید با ۱ شروع بشود. (طبق گفته هد درس) حال ماشین آن را به شکل زیر پیاده‌سازی می‌نماییم.



یک شبیه سازی برای ورودی ۱۱۱$۱۱ که عضو زبان است را انجام می‌دهیم.

۱۱۱$۱۱
X۱۱$۱۱
X۱۱$۱۰
XX۱$۱۰
XX۱$۰۱
XXX$۰۱
XXX$۰۰

همچنین روی دو نمونه دیگر شبیه‌سازی و تست انجام شده که تصاویر آن به فایل مربوطه پیوست شده است.

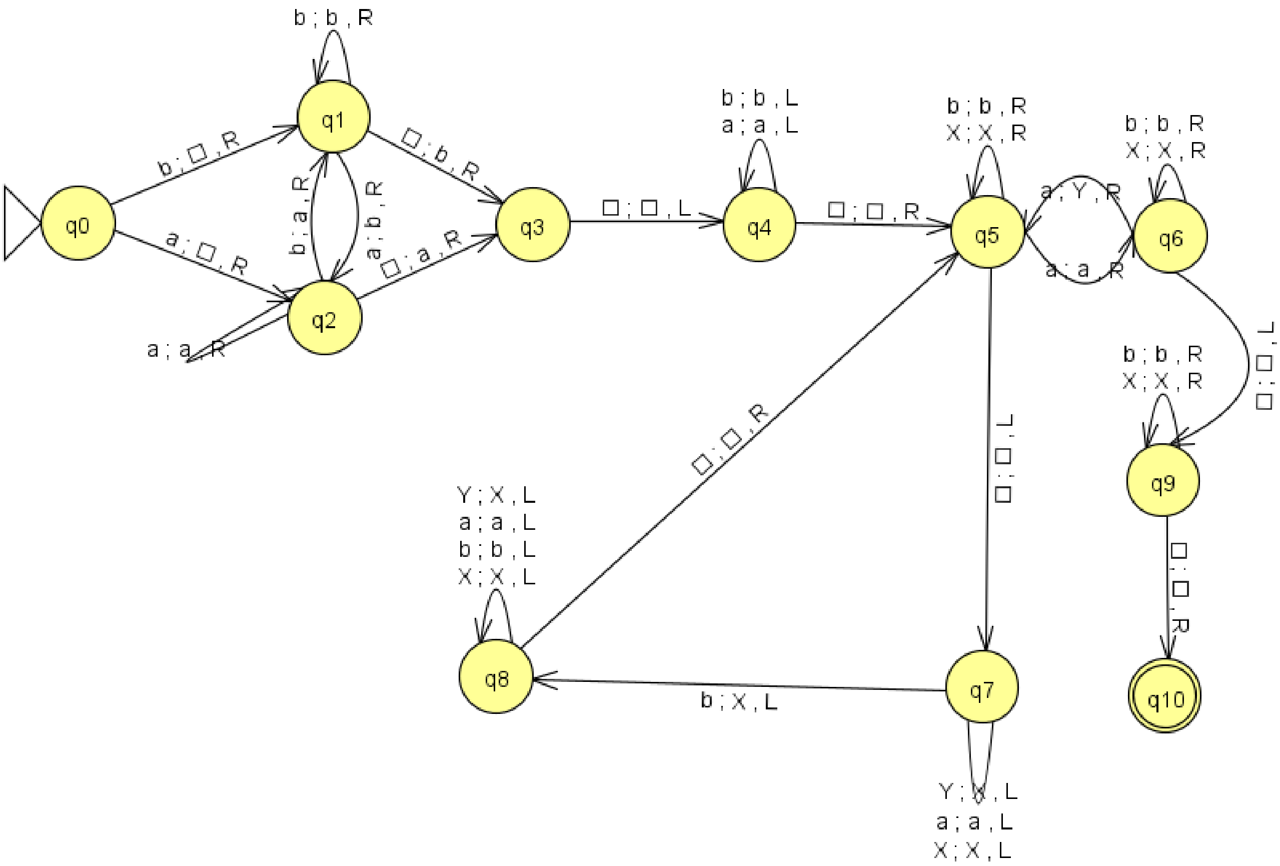
ج)

کمی سخت است اما سعی کردم که این بخش را هم با ماشین تک‌نواره پیاده‌سازی نمایم. ابتدا در مراحلی در ابتدای رشته، عبارت blank را قرار می‌دهیم. البته در jflap خودش این کار را انجام می‌دهد ولی برای اینکه مطابق اسلایدها پیش برویم این کار را انجام می‌دهیم. دقت کنید حتما باید یا دقیقا یک عدد a و یا زوج تا از آن داشته باشیم. ایده ما این است که ابتدا نصف تعداد هر a، آن‌ها را مارک می‌کنیم. سپس از آخر شروع کرده و به ازای یک b، تمامی این مارک شده‌ها را با X عوض می‌کنیم. برای مثال در نظر بگیرید که در ابتدا ۸ عدد a داریم. در مرحله اول ۴ عدد a با یک عدد b تبدیل به X می‌شوند. حال ۴ عدد a باقی مانده است. مجددا دو عدد a و یک عدد b تبدیل به X می‌شوند. حال ۲ عدد a مانده است که یکی از آن‌ها همراه تنها b باقی‌مانده از

بین می‌روند و و تنها یک عدد a می‌ماند که در این صورت اگر دیگر b نداشتیم، اکسپت می‌شود.
مثال برای ورودی $aabaabaabaa$:

$$aabaabaabaa$$
$$\square aabaabaabaa$$
$$\square aYbaabaabaa$$
$$\square aYbaYbaabaa$$
$$\square aYbaYbaYbaa$$
$$\square aYbaYbaYbaY$$
$$\square aYbaYbaYbaX$$
$$\square aYbaYbaYXaX$$
$$\square aYbaYbaXXaX$$
$$\square aYbaXbaXXaX$$
$$\square aXbaXbaXXaX$$
$$\square aXbYXbaXXaX$$
$$\square aXbYXbaXXYX$$
$$\square aXbYXbaXXXX$$
$$\square aXbYXXaXXXX$$
$$\square aXbXXXaXXXX$$
$$\square aXbXXXYXXXX$$
$$\square aXbXXXXXXXX$$
$$\square aXXXXXXXXXX$$

حال شبیه سازی آن را در jflap به نمایش می‌گذاریم. البته توجه کنید همانطور گفتیم خود jflap به طور دیفالت قبل و بعد رشته، blank می‌گذارد. اما ما در این بخش آن مرحله blank گذاری اولیه را نیز انجام می‌دهیم.



۳ تست دیگر نیز انجام شده است که نتایج آن در تصاویر کنار این فایل قرار دارند.

۲. پرسش دوم

پاسخ.

۱)

ابتدا یک تعریف اولیه برای خودکاره ۲ پشته‌ای می‌آوریم.

A *2-PDA* is a PDA with two stacks instead of one. It behaves like a normal PDA, except that a 2-PDA can pop from and push to either, both or neither of the stacks. The transition function of a 2-PDA is defined as

$$\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \times \Gamma_\epsilon \to Q \times \Gamma_\epsilon \times \Gamma_\epsilon.$$

حال توصیف کامل آن را داریم:

یک خودکاره ۲ پشته‌ای، یک $\text{۷} - tuple$ است که به شکل زیر توصیف می‌شود:

$$(Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, F)$$

- Q is the set of states.
- Σ is the input alphabet.
- Γ_1 is the alphabet of stack 1.
- Γ_2 is the alphabet of stack 2.
- $\delta: Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma_1 \cup \{\epsilon\}) \times (\Gamma_2 \cup \{\epsilon\}) \to \mathcal{P}(Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma_1 \cup \{\epsilon\}) \times (\Gamma_2 \cup \{\epsilon\}))$
- $q0 \in$ Q is the start state.
- F ⊆ Q is a set of accept states.

دقت کنید در صورت سوال به قطعی یا عدم قطعی بودن اشاره‌ای نشده است. ما حالت غیرقطعی که کلی‌تر هست رو نشان می‌دهیم. همچنین شرایط پذیرش رشته توسط این خودکاره به شکل زیر است:

A 2-NPDA $M = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, q_0, F)$ accepts string $w \in \Sigma^*$ if w can be written as

$$w_1, w_2, \ldots, w_m \in (\Sigma \cup \{\epsilon\})^*,$$

and there exist states

$$r_0, r_1, \ldots r_m$$

and pairs of strings

$$(s_0, t_0), (s_1, t_1), \ldots, (s_m, t_m) \in (\Gamma_1 \cup \{\epsilon\})^* \times (\Gamma_2 \cup \{\epsilon\})^*$$

such that

- $r_0 = q_0$, and
- $(s_0, t_0) = (\epsilon, \epsilon)$, and
- $(r_{i+1}, c, d) \in \delta(r_i, w_{i+1}, a, b)$, where $s_i = au$ and $s_{i+1} = cu$ for some $u \in \Gamma_1^*$, and $t_i = bv$ and $t_{i+1} = dv$ for some $v \in \Gamma_2^*$, and
- $r_m \in F$.

حال باید نشان دهیم که هم‌ارز با رایانه تورینگ است.

برای اثبات معادل بودن دو نوع ماشین، باید نشان دهیم که با توجه به هر ماشین تورینگ، می‌توانیم یک خودکاره ۲ پشته‌ای معادل بسازیم و بالعکس.

ابتدا ماشین تورینگ را با این خودکاره شبیه سازی می کنیم. ایده اصلی استفاده از S۱ (اولین پشته) برای نشان دادن همه چیز در سمت چپ head ماشین تورینگ است. از S۲ (پشته دیگر) برای نشان دادن نقطه زیر head و همه چیز در سمت راست آن استفاده می‌کنیم. خواندن معادل popping از S۲ و نوشتن معادل pushing نماد جدید روی S۲ است. حرکت به راست معادل popping از S۲ و pushing نماد پاپ شده روی S۱ است. حرکت به چپ معکوس حرکت به راست است.

ما باید خودکاره را مقداردهی اولیه کنیم. در ماشین تورینگ، هد از ابتدای رشته ورودی شروع می شود. در نمایش خودکاره ما، این همان رشته ورودی است که در S۲ با شروع رشته در بالای پشته قرار دارد. برای بدست آوردن این، رشته ورودی را می خوانیم و هر نماد را روی S۱ پوش می‌کنیم. سپس هر نماد را از S۱ پاپ کرده و آن را روی S۲ پوش می‌کنیم تا نقطه شروع و جهت رشته مورد نظر را بدست آوریم. اکنون می توانیم اجرای ماشین تورینگ شبیه سازی شده را آغاز کنیم. نکته دیگری که باید به آن توجه کرد این است که ماشین تورینگ می تواند در طول دلخواه نوار به سمت چپ یا راست رشته ورودی حرکت کند. این می تواند منجر به یک پشته خالی پس از حرکت های کافی شود. بنابراین اگر در حین حرکت، یک پاپ از پشته نشان دهنده یک پشته خالی باشد، برای حفظ موقعیت head، یک نماد خالی را روی پشته دیگر پوش می کنیم.

اکنون نشان خواهیم داد که با توجه به $NPDA - ۲$، می‌توانیم یک ماشین تورینگ معادل بسازیم. از اسلایدها

می دانیم که ماشین های تورینگ چند نواری (غیر قطعی) معادل ماشین های تورینگ نواری هستند. بنابراین، می‌توانیم یک ماشین تورینگ غیر قطعی ۳ نواری بسازیم، که در آن نوار ورودی با نوار ورودی $NPDA-۲$ یکسان است و دو نوار کار دو پشته $NPDA-۲$ را شبیه‌سازی می‌کنند.

نوارهای موجود در ماشین تورینگ را $T۰$، $T۱$، $T۲$ و پشته‌های موجود در $NPDA-۲$ $S۱$ و $S۲$ را می‌نامیم. برای شبیه سازی پایین پشته های $NPDA-۲$ روی هر یک از نوارهای کاری یک نماد خاص می‌نویسیم. برای شبیه سازی پوش کردن a به $S۱$، به سمت راست در $T۱$ حرکت می کنیم و سپس a را می نویسیم. برای شبیه سازی پوش کردن a به $S۲$، به سمت راست در $T۲$ حرکت می کنیم و سپس a را می نویسیم. برای پاپ از $S۱$ از $T۱$ بخوانید و به سمت چپ حرکت می‌کنیم. اگر خوانده شده نماد ویژه را نشان دهد، به انتهای پشته رسیده ایم بنابراین به سمت چپ حرکت نمی کنیم. پاپ از $S۲$ یکسان است. حال شبیه‌سازی را انجام دادیم. ما استفاده از دو پشته $NPDA-۲$ را شبیه سازی کرده ایم، بنابراین می توانیم اجرای $NPDA-۲$ را به طور کامل شبیه سازی کنیم. نکته مهم این است که ماشین شبیه‌سازی غیرقطعی است، بنابراین هر یک از چندین "مرحله بعدی" احتمالی $NPDA-۲$ به درستی (در مسیرهای محاسباتی غیر قطعی جدا شده) توسط TM با استفاده از غیر قطعی بودن آن شبیه‌سازی شده است.

دقت کنید اشاره کردیم که حالت کلی‌تر $NPDA-۲$ را در این بخش نشان دادیم. برای حالت عادی مانند راهی که در تصویر زیر وجود دارد عمل می‌کنیم.

To simulate a 2-PDA on a Turing machine, we construct a Turing machine that stores the configuration of the PDA. It then performs a BFS on the possible configurations of the PDA, and accepts iff it finds a valid path to an accepting configuration.

To simulate a Turing machine on a 2-PDA, we first ensure that the 2-PDA is deterministic. We use the right stack to represent the tape after (or at) the Turing machine's head, and the left stack to represent the tape before the Turing machine's head. We thus must first move the input into the right stack. Once that is done, we can determine all our actions by reading from the right stack, and perform moves by moving a character from the right stack to the left stack or vice versa.

We simulate a 2-PDA on a Turing machine by using the tape as a "queue" of possible configuration of the 2-PDA (say by simply keeping them in sequence with delimiters, popping by reading the first configuration and then moving the others left, and pushing by adding configurations at the end). Each configuration in the "queue" encodes a state of the 2-PDA, the contents of both stacks, and the remaining input. Our TM will then pop a configuration, iterate through the possible 2-PDA transitions from this configuration, and for each transition pushing the resulting configuration onto the stack. If the transition popped has no more input and is in an accept state, the TM accepts. If the "queue" is ever empty, the TM rejects.

The 2-PDA accepts iff there is some sequence of valid transitions which brings it to an accept state with no input left. If such a sequence exists the TM will eventually follow it and accept; conversely, if no such sequence exists the TM will not accept. Thus this TM accepts iff the 2-PDA did.

Conversely, we simulate a TM $T = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ with a 2-PDA $P = (Q \sqcup (Q'\Sigma, \Gamma \sqcup \{L\}, \delta', q_0', \{q_a\})$ with

$$Q' = Q \sqcup \{r(q,g), l(q,g) \mid q \in Q, g \in \Gamma\} \sqcup \{q_0', q_L, q_R\}$$

and δ' as follows:

$$q'(q,c,g_l,g_r) = \begin{cases} \{(q_R, L, \varepsilon)\}, & q = q_0', c = g_l = g_r = \varepsilon \\ \{(q_R, \varepsilon, c)\}, & q = q_R, c \neq \Box, g_l = g_r = \varepsilon \\ \{(q_L, \varepsilon, \varepsilon)\}, & q = q_R, c = \Box, g_l = g_r = \varepsilon \\ \{(q_L, \varepsilon, g_r)\}, & q = q_L, c \neq L, c = g_r = \varepsilon \\ \{(q_0, \varepsilon, \varepsilon)\}, & q = q_L, g_l = L, c = g_r = varepsilon \\ \{(l(q', c'), g_l, \varepsilon)\}, & \delta(q, g_r) = (q', c', R), g_l = c = \varepsilon \\ \{(r(q', g_l), \varepsilon, c')\}, & \delta(q, g_r) = (q', c', L), g_l = c = \varepsilon \\ \{(q', c', \varepsilon)\}, & q = l(g', c'), g_l = g_r = \varepsilon \\ \{(q', \varepsilon, c')\}, & q = r(g', c'), g_l = g_r = \varepsilon \\ \{\}, & \text{otherwise.} \end{cases}$$

All outputs of δ' are size 1 or 0, so P is deterministic. It first adds the entire input to the first stack, then reverses it so it is in the second stack with the first character at the top. At any point, it will modify the tape to the left and right of the head exactly as T would, ensuring that it will ever reach an accept state iff T will ever reach q_a. Thus it accepts exactly the strings T does.

۲)

در این بخش با خودکاره صفی کار داریم. تعریف آن را هم در صورت سوال و هم در بخش زیر مشاهده می‌کنید.

Here is an informal description of a Queue Automaton: the machine is similar to an NPDA, except that the stack ("last in, first out" memory) is replaced with a queue ("first in, first out" memory). At each step, the machine reads a symbol from the tape (possible ϵ), dequeues a specified symbol (possibly ϵ) from the head of the queue, enqueues a specified symbol onto the tail of the queue (possibly ϵ), and moves into a specified state. The machine accepts if there is some computation on its input string that causes it to reach an accept state.

یک خودکاره صفی، یک $tuple - ۶$ است که به شکل زیر توصیف می‌شود:

$$(Q, \Sigma, \Gamma\delta, q_., F)$$

- Q is the set of states.

- Σ is the input alphabet.
- Γ is the queue alphabet.
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\epsilon\}))$
- $q0 \in$ Q is the start state.
- F $\subseteq$ Q is a set of accept states.

همچنین شرایط پذیرش رشته توسط این خودکاره به شکل زیر است:

$$w_1, w_2, \ldots, w_m \in (\Sigma \cup \{\epsilon\})^*,$$

and there exist states

$$r_0, r_1, \ldots r_m$$

and strings

$$s_0, s_1, \ldots, s_m \in (\Gamma \cup \{\epsilon\})^*$$

such that

- $r_0 = q_0$, and
- $s_0 = \epsilon$, and
- $(r_{i+1}, c) \in \delta(r_i, w_{i+1}, a)$, where $s_i = au$ and $s_{i+1} = uc$ for some $u \in \Gamma_1^*$,
- $r_m \in F$.

ابتدا، نحوه شبیه سازی ماشین تورینگ با خودکار صفی (QA) را نشان می دهیم. صف ما همیشه حاوی محتویات نوار ماشین تورینگ خواهد بود و نماد در حال حاضر در بالای صف خوانده می شود. ما از "$" برای علامت گذاری ابتدای نوار استفاده می کنیم. بنابراین همیشه صف اینگونه به نظر می رسد:

$$ax\$y$$

$a \in \Sigma$ is the symbol currently under the head of the Turing Machine
$x \in \Sigma^*$ is the contents of the tape to the right of the head
$y \in \Sigma^*$ is the contents of the tape to the left of the head.

ما یک صف "ابتدایی" را توصیف خواهیم کرد که در شبیه سازی TM از آن استفاده خواهیم کرد

حال عملیات Cyclically shift right را تعریف می‌کنیم. این عملیات را در سه فاز انجام داده و فرض می‌کنیم که محتوای صف به شکل $ab\$c$ باشد.

فاز اول.

یک نشانگر # در انتهای صف قرار می دهیم و سپس هر علامت صف x را با یک نماد جدید (w, x) جایگزین می کنیم. که w سیمبلی است که بلافاصله سمت چپ x آمده است. برای این از مجموعه ای از استیت های جدا از هم استفاده می‌نیم که اخرین سمبلی که شیفت خورده است را به یاد می آورند. بنابراین برای هر $w \in \Sigma$، بعد از شیفت خوردن w در استیت q_w هستیم. سپس هنگامی که با نماد بعدی x مواجه می شویم، نه x، بلکه نماد جدید (w, x) را در صف قرار می دهیم.

برای شروع فرآیند، # را در صف قرار می دهیم و به حالت $q_\#$ می رویم. سپس موارد زیر را تکرار می کنیم: از یک حالت داده شده q_w، نماد x را از صف خارج کنید، نماد جدید (w, x) را در صف قرار دهید و به حالت q_x بروید. وقتی # را از صف خارج می‌کنیم و سمبل جدید نهایی را در صف قرار می دهیم فاز ۱ را کامل می کنیم. برای مثال در رشته گفته شده داریم:

$$(\#, a)(a, b)(b, \$)(\$, c)(c, \#)$$

فاز دوم.

گام زیر را تکرار می‌کنیم: به طور مداوم (w,x) خارج و داخل صف می‌کنیم تا زمانی که $x = \#$ بشود. آن‌گاه ابتدا # و سپس w را وارد صف می‌کنیم. سپس هر سمبلی که head روی آن است است را خارج می‌کنیم. برای همان مثال داریم:

$$(a,b)(b,\$)(\$,c)\#c$$

فاز سوم.

در این فاز همه (w,x)ها را خارج می‌کنیم و w را وارد صف می‌کنیم تا زمانی که به # برسیم. این فاز در همین مرحله تمام می‌شود. برای مثال خود داریم:

$$cab\$$$

حال اگر دقت کنید مشاهده می‌شود خروجی این ۳ فاز، همان صف ورودی اولیه است که یک دور به طور چرخشی شیفت راست خورده است.

حال برای شبیه سازی یک مرحله معین از ماشین تورینگ، نماد a را از بالای صف خارج می‌کنیم و سپس داریم:

- اگر ماشین تورینگ با خواندن a کاراکتر b را مینویسد و به سمت راست حرکت می‌کند آنگاه در خودکاره b را در اخر صف می‌نویسیم.
- مشابه حالت بالا اگر به چپ حرکت کند، b را وارد می‌کنیم و سپس دو شیفت چرخشی به راست خواهیم داشت.

در تمام نقاط، توالی انتقال زیر برای QA در دسترس است:

$ را از صف خارج می‌کنیم. $-$ را وارد صف می‌کنیم. و ۲ بار cyclic shifts به راست خواهیم داشت. که این به معنی اضافه کردن blank به آخر tape می‌باشد. حال ما مقداردهی خودکاره صفی را با کپی کردن ورودی در صف به همراه $ در ته آن خواهیم داشت.

در نهایت، باید نشان بدهیم چگونه یک صفی را با رایانه تورینگ شبیه سازی می‌کنیم. از آنجایی که رایناه تورینگ ۲ tape عه با حالت single tape معادل است، نشان دادن نحوه شبیه سازی QA با TM ۲ نواری کافی است. نوار اول به سادگی شامل رشته ورودی است و نوار دوم شامل صف خواهد بود. الفبای نوار QA حاوی نماد ویژه $ است که شروع صف را نشان می دهد. ابتدا یک $ روی نوار مربوط به صف نوشته می شود. هنگامی که یک نماد به صف پوش می‌شود، هد اولین فضای خالی روی نوار را پیدا می‌کند و نماد را در آنجا می‌نویسد. هنگامی که یک نماد پاپ می‌شود، نوار اولین نماد روی نوار غیر از $ را پیدا می کند، نماد را می‌خواند و نماد را با $ جایگزین می‌کند. با پاپ کردن TM صف روی نوار به سمت راست شیفت می‌کند، اما این قابل قبول است زیرا نوار بی نهایت است. بنابراین هم‌ارزی این دو ماشین اثبات شد.

برای حل این سوال از این لینک کمک گرفته شده است.

۳. پرسش سوم

پاسخ.

۱.

ایده این سوال از هد پرسیده شده است.

ماشین تورینگ M را برای این زبان در نظر بگیرید. این ماشین بدین گونه عمل میکند که به ازای هر $q \in Q$ و هر $a \in \Sigma \cup \epsilon$ و هر $b \in \Gamma$ چک میکند که آیا $\delta(q, a, b)$ حداکثر یک عضو دارد یا خیر.

اگر برای $a = \epsilon$ حاصل دلتا دقیقا یک عضو داشت، آنگاه حاصل دلتا به ازای همه آلفابت برای a، تهی باشد. این از شروط خودکاره پشته ای قطعی می باشد.

این عملیات در زمان متناهی قابل انجام است. ماشین تورینگ M حتما halt میکند و زبان خواسته شده را تصمیم میگیرد (recognizer) . بنابراین زبان گفته شده، تصمیم پذیر است.

۲.

سعی می‌کنیم این زبان را به زبان معادل تبدیل کنیم.

ابتدا فرض کنید زبان A داریم که $L(D) = A$. در واقع A زبان منظمی است که ماشین قطعی D آن را تشخیص می‌دهد. می‌دانیم زبان‌های منظم تحت ریورس بسته هستند. بنابراین زبان A^R نیز منظم است و یک ماشین قطعی دیگری مانند M وجود دارد که $L(M) = A^R$. حال دقت کنید زبان داده شده در این بخش با زبان زیر معادل است:

$$L = \{< D > \ | L(D) = L(M)\}$$

حال از اسلایدهای درس می‌دانیم که EQ_{DFA} تصمیم‌پذیر است. فرض کنید ماشین تورینگ E، یک decider برای EQ_{DFA} است. برای اینکه نشان دهیم زبان L تصمیم‌پذیر است، یک ماشین تورینگ Q به شکل زیر معرفی می‌کنیم:

Q : " on input <D>, where D is a DFA:
1. Construct M which is reverse of D
2. Run TM E on input <D, M>
3. if E accepts, accept. If E rejects, reject. "

می‌دانیم که E یک decider است. بنابراین Q نیز برای زبان L یک decider خواهد بود. در نتیجه این زبان تصمیم‌پذیر است.

۳.

ایده سوالات زیرمجوعه را به این شکل انجام می‌دهیم: اینگونه زبان‌ها معادل با این هستند که اشتراک اولی با مکمل دومی تهی است. بنابراین باید ببینیم زبان زیر تصمیم‌پذیر هست یا خیر:

$$L = \{< G, D > \ | L(G) \cap \overline{L(D)} = \emptyset\}$$

در نظر داشته باشید که اشتراک زبان منظم با زبان مستقل از متن، مستقل از متن است. بنابراین یک دستور زبان مستقل از متن U برای زبان $L(G) \cap \overline{L(D)}$ داریم. همچنین در اسلایدها نشان دادیم که E_{CFG}، تصمیم‌پذیر است. فرض کنید S در واقع decider آن است. حال یک ماشین تورینگ K را به عنوان decider زبان مورد نظر نشان می‌دهیم. داریم:

K : ” on input <G,D>, where G is a CFG and D is a DFA:
1. Construct U
2. Run TM S on input U
3. If S accepts, accept. If S rejects, reject.

با توجه به تصمیم گیرنده بودن S، قطعا K هم decidable خواهد بود و زبان این بخش، تصمیم‌پذیر است.

۴. پرسش چهارم

پاسخ.

۱.

تحت اجتماع بسته هستند.

دو ماشین تورینگ M_1 و M_2 را در نظر بگیرید. ماشین تورینگ M را به نوعی می‌سازیم تا اجتماع زبان این دو ماشین را تشخیص بدهند. داریم:

M : on input w:
1. Run both TM M1 and TM M2 in parallel
2. If either accepts, accept. If both halt and reject, reject.

همانطور که می‌بینید ماشین تورینگ بالا برای اجتماع این دو زبان تشخیص پذیر است . توجه داشته باشید اگر هیچ کدام اکسپت نکنند و حداقل یکی از آنها در loop بیفتد، ماشین M نیز با looping رد خواهد کرد. (اکسپت نخواهد کرد.) بنابراین تحت اجتماع بسته هستند.

۲.

تحت متمم بسته نمی‌باشند.

مثال نقض این مورد را در اسلایدها داشته‌ایم. A_{tm} تشخیص‌پذیر است اما $\overline{A_{tm}}$ حتی تشخیص‌پذیر هم نیست.

۳.

تحت هم‌ریختی بسته هستند.

فرض کنید زبان L تشخیص پذیر است و ماشین تورینگ آن نیز M است. می‌خواهیم ماشین تورینگ N را طوری برای زبان $h(L)$ بسازیم که:

$$h(L) = \{h(w) \mid w \in L\}$$

حال ترتیب lexicographic زبان شمارای Σ^* را در نظر بگیرید. فرض می‌کنیم رشته‌های این زبان به ترتیب $w_1, w_2, \ldots$ باشند. حال ماشین N را می‌سازیم:

N = on input w:
1. Let A = $\emptyset$
2. Repeat the following for i = 1, 2, 3, . . ., if h(w_i) = w:
3. A = A $\cup$ w_i

4. Run M for |A| steps on all the strings in A.
5. If any accepts, accept.

بنابراین اگر رشته‌ای عضو L باشد که $w = h(w_i)$، آن گاه N آن را در زمان متناهی اکسپت می‌کند. بنابراین $h(L)$ تشخیص پذیر است و اثبات شد.

۴.

تحت بستار چرخشی بسته هستند.

در نظر بگیرید ماشین تورینگ M زبان L را می‌پذیرد. سپس ماشین M' را در نظر بگیرید:

۱. به ازای ورودی $w = w_۱w_۲ \dots w_n$ به گونه غیرقطعی ، واک $w_i,\ ۱ \leq i \leq n$ را برمی‌گزیند و رشته ی بر روی نوار به شکل $w^i = w_i w_{i+۱} \dots w_{i-۱}$ در می‌آورد.

۲. سپس ماشین M را بر روی w^i پیش می‌برد.

اکنون نشان می‌دهیم که M' زبان $RC(L)$ را می‌پذیرد.

- اگر $w \in RC(L)$ باشد آنگاه x, y به گونه ای یافت می‌شود که $w = xy$ $xy \in L,$. اگر $x = w_i \dots w_n$ باشد، آنگاه با گزیدن i ماشین M'، رشته xy را بر روی نوار درست می‌کند. سپس از آنجا که M پذیرنده ای برای زبان L است، نتیجه میشود که مسیر محاسباتی پیدا میشود که در آن w به دست M' پذیرفته شود.
- اگر هم w عضو $RC(L)$ نباشد، آنگاه با هیچ چرخشی نمیتوان آن را به یک رشته در L تبدیل کرد. بنابراین M' هرگز رشته w را نمیپذیرد.

۵.

این هم بسته است. از بخش ۳ کمک می‌گیریم.

همانند بخش های قبل ماشین تورینگ M و زبان L را در نظر بگیرید. ماشین تورینگ N را می‌سازیم که:

$$L(N) = h^{-۱}(L) = \{w \mid h(w) \in L\}$$

داریم:

N : on input w:
1. Compute h(w)
2. Run TM M on h(w)
3. If M accepts, accept.

همانطور که مشخص است، اگر M اکسپت کند یعنی $h(w)$ عضو L بوده است پس w نیز عضو وارون همریختی خواهد بود. پس اثبات شد.

۵. پرسش پنجم

پاسخ.

۱.

باید نشان دهیم زبان‌ها، undecidable هستند.

آ)

با برهان خلف فرض می‌کنیم که decidable باشد. بنابراین ماشین تورینگ S وجود دارد که decider آن است.

از طرفی طبق اسلایدها می‌دانیم E_{tm} تصمیم‌ناپذیر است. حال ماشین تورینگی برای E_{tm} می‌سازیم که ورودی یک ماشین می‌گیرد. سپس یک ماشینی می‌سازیم تا تمامی رشته‌ها را رجکت نماید. حال ماشین تورینگ S را اجرا کرده و این دو ماشین را به آن می‌دهیم. اگر اکسپت کند، یعنی زبان ماشین ورودی، زیرمجموعه ماشین ساخته شده است و در واقع تهی است. پس ماشین E_{tm} نیز آن را اکسپت می‌کند و بالعکس. پس نتیجه می‌شود که E_{tm} تصمیم‌پذیر است که این یک تناقض است.

N : on input M which M is TM:
1. Construct TM U that rejects all strings
2. Run TM S on input <M, U>
3. If S accepts, accept. If S rejects, reject.

بنابراین $SUBSET_{tm}$ تصمیم‌ناپذیر است.

ب)

این سوال، سوال اول فصل ۵ کتاب سیپسر می‌باشد.

Proof to show that EQ_{CFG} is undecidable:

Step-1:

Consider a context-free grammar CFG $G_0 = (V, \Sigma, R, S)$ where $V = \{S\}$ and S is a starting variable. Assume that there is a rule $S \rightarrow lS$ in R for every terminal $l \in \Sigma$. The grammar G_0 includes a $\in$ notation by using the rule $S \rightarrow \in$.

Example:

For the CFG, the rules in G_0 are defined as $S \rightarrow aS \mid bS \mid \in$ over the alphabet set $\Sigma = \{a, b\}$. So, the grammar CFG G_0 satisfies all the alphabets in the alphabet set Σ.

So, $L(G_0) = \Sigma^*$. Thus, the Turing Machine is decidable.

Step-2:

Assume that the CFG is decidable by using the Turing machine R that decides EQ_{CFG}. Construct another Turing machine S which uses R to decide ALL_{CFG} by using the following procedure:

$S = On\ input\ \langle G_0 \rangle,$

1. Run *R* on the input $\langle G_0, G_1 \rangle$. G_1 is a CFG, which generates Σ^*.
2. Accept the grammar, when *R* accepts.
3. Otherwise reject.

Thus, if the Turing machine *R* decides EQ_{CFG}, *S* also decides ALL_{CFG} which is impossible. So, EQ_{CFG} is also undecidable.

همانطور که می‌بینید به کمک تصمیم‌ناپذیری ALL_{CFG} که در اسلایدها نشان دادیم، اثبات می‌شود.

ج)

دقیقا مشابه بخش قبل، از ALL_{CFG} کمک می‌گیریم.

طبق برهان خلف فرض کنید L تشخیص پذیر است و ماشین تورینگ M آن را تشخیص می‌دهد. حال ماشین تورینگ N برای ALL_{CFG} را به شیوه زیر در نظر بگیرید:

N : on input <G>, which G is a CFG:
1. Construct the DFA D which L(D) = Σ^*
2. Run TM M on <G, D>
3. If M accepts, accept. If M rejects, reject.

بنابراین اگر ماشین M بگوید که آن CFG ورودی و ماشین قطعی ساخته شده با هم برابر هستند، در نتیجه ماشین N نیز باید اکسپت شود زیرا CFG ورودی، تمامی رشته‌ها را داراست. بنابراین باید M هم تصمیم‌پذیر باشد که این یک تناقض است.

۲.

آ)

ابتدا فرض کنیم K تشخیص‌پذیر است و ماشین تورینگ S آن را تشخیص می‌دهد. حال ماشین زیر را می‌سازیم:

D = on input M:
1. accept if S(<M>) rejected.
2. reject if S(<M>) accepted.

حال دقت کنید که به ماشین D خودش را ورودی می‌دهیم. داریم:

1. If <D> $\in$ L(D), D rejects <D>.
2. If <D> $\notin$ L(D), D accepts <D>.

که این تناقض است و به دلیل فرض تشخیص‌پذیر بودن K حاصل شده است. بنابراین این زبان، تشخیص‌ناپذیر است.

راه کامل‌تر:
همان فرض خلف را داریم. ماشین را به شکل زیر میسازیم:

D = on input M where M is TM:
1. Run M and S on input <M> in parallel
3. If M accepts, reject. If S accepts, accept.

بنابراین به همان تناقض رسیدیم.

ب)

می‌دانیم از بخش قبل که زبان K تشخیص‌ناپذیر است. حال برای اینکه ثابت کنیم EQ_{tm} نیز تشخیص‌ناپذیر است، زبان K را به آن کاهش می‌دهیم.

برای این کار مشابه اسلایدها از mapping reducibility استفاده می‌کنیم.

ماشین تورینگ برای تابع کاهش K به EQ_{tm} را مطابق زیر می‌نویسیم:

Z = on input <M>, where M is a TM:
1. Construct:
2. M1 = rejects on any input
3. M2 = on any input: Run M on input <M>. If M accepts, accept. If M rejects, reject.
4. Output <M1, M2>.

می‌دانیم وقتی ماشینی عضو K هست یعنی خودش را رجکت می‌کند. این ماشین یک ورودی از K می‌گیرد و یک خروجی برای EQ می‌دهد (به طور شهودی) . حال، ما ۲ ماشین می‌سازیم. $M۱$ که همه چیز را رجکت می‌کند و $M۲$ که $<M>$ را به آن می‌دهیم. اگر M خودش را رجکت کند، زبان $M۲$ تهی است. پس زبان $M۱$ و $M۲$ یکی هست و به EQ_{tm} می‌رسیم. پس کاهش ما درست است و این زبان، تشخیص‌پذیر نمی‌باشد.

ج)

همانطور که گفته شده است، زبان EQ_{tm} را به این زبان کاهش می‌دهیم.
برای این کار مشابه اسلایدها از mapping reducibility استفاده می‌کنیم.
داریم:

Z = on input <M1, M2>, which M1, M2 are TMs:
1. Construct the M3 which:
2. M3 = only accepts ϵ
3. Output <M1, M2, M3>.

در واقع $M۳$ را برابر با ϵ قرار می‌دهیم و کاهش گفته شده درست خواهد بود. زیرا:

$$L(M۱) = L(M۲)L(M۳) = L(M۲)\,\epsilon = L(M۲)$$

۶. **پرسش امتیازی**

پاسخ.
جواب این سوال در فایل $TFLA\_HW۴\_Q۶.pdf$ ضمیمه شده موجود می‌باشد.

- از هد درس برای لاتک نکردن سوال امتیازی اجازه گرفته شده است.
- برای حل این سوال از این لینک کمک گرفته شده است.

موفق باشید :)