تمرین سری هفتم

درس نظریه‌ی زبان‌ها و ماشین‌ها

آیدین نصیری‌شرق ۸۳۱۱۳۸۰۴

مسأله‌ی ۱:

توجّه: فرض شده که در صورت سؤال، منظور از $w \in \{0, 1\}^*$، $w \in \{a, b\}^*$ بوده است.

یک راه ساده برای ساختن ماشین تورینگ تصمیم‌پذیر برای این زبان‌ها، این است که ابتدا a ها و b های آن‌ها را در انتهای رشته‌ی ورودی طوری مرتّب کنیم که ورودی اوّلیّه تماماً به x تبدیل شده و در انتهای آن ابتدا یک عنصر # و پس از آن، ابتدا تمامی a ها و سپس تمامی b ها بیاید. به عنوان مثال، ورودی $abbaa$ روی نوار، به رشته‌ی $xxxxx\#aaabb$ تبدیل شود.

برای این منظور، کافی‌ست مطابق رویه‌ی زیر عمل کنیم:

- در مرحله‌ی صفرم، در انتهای ورودی (جایی که اوّلین عنصر خالی یافت می‌شود)، یک عنصر # درج کرده و به ابتدای نوار برگرد.
- پس از آن، در مرحله‌ی اوّل، با شروع از ابتدای نوار، با دیدن هر a، آن را به x تبدیل کرده و تا انتهای نوار می‌رویم، سپس در آن‌جا یک a نوشته و در پایان مجدداً به ابتدای نوار برمی‌گردیم. این کار را تا موقعی که هیچ aیی در نوار قبل از # باقی نماند (و هنگام جستجو برای یافتن a، به # رسیدیم) انجام می‌شود.
- سپس در مرحله‌ی دوم نیز، همان کارِ مرحله‌ی قبل را برای b ها انجام می‌دهیم.

پس از این تبدیلات، کافی‌ست رشته‌ی جاری را (با در نظر نگرفتن x های ابتدای رشته و # پس ازآن‌ها) به ماشین تصمیم‌گیرنده‌ی $a^n b^n$ که در کتاب درس توضیح داده شده ارجاع داده و نتیجه‌ی آن را به عنوان نتیجه‌ی ماشین مربوطه برگردانیم. تصمیم گیرنده‌ی $a^n b^{2n}$ نیز قواعدی مشابه دارد.

امّا راه‌حل مستقیم مسأله بدین شرح است:

الف) ایده‌ی اصلی نحوه‌ی کار ماشین چنین است که از ابتدای رشته شروع کرده و به دنبال یک عنصر a یا b می‌گردد (در حالت q_0)؛

- در صورتی که چنین عنصری یافت نشد، خروجی پذیرش[1] را بر می‌گرداند؛

[1] $accept$

- امّا در صورتی که چنین عنصری یافت شد، آن را مارک (تبدیل به x کرده) و به سمت جلو حرکت می‌کند تا یک عنصر مخالف آن را پیدا کند (به ترتیب q_1 برای a ی یافت شده و q_2 برای b ی یافت شده):

—— در صورت پیدا نشدن آن عنصر مخالف (یعنی a، وقتی در مرحله‌ی قبل یک b را تبدیل به x کردیم و b، برای وقتی که در مرحله‌ی قبل یک a را به x تبدیل کردیم) ، نتیجه‌ی رَد[2] را بر می‌گرداند؛

—— امّا در صورت پیدا شدن، آن عنصر را نیز مارک (تبدیل به x کرده)، به ابتدای نوار برگشته (تحت q_2) و مجدّداً همین رویه را در پیش می‌گیرد.

توصیف رسمی[3] این ماشین نیز چنین است:

$$
\begin{aligned}
&M_1 = \{Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}\}\\
&Q = \{q_0, q_1, q_2, q_3, q_{accept}, q_{reject}\},\\
&\Sigma = \{a, b\},\\
&\Gamma = \{a, b, x\},\\
&\delta = \{(q_0, \sqcup) \to (q_{accept}, \sqcup, R), (q_0, a) \to (q_1, x, R), (q_0, b) \to (q_3, x, R), (q_0, x) \to (q_0, x, R),\\
&\quad (q_1, \sqcup) \to (q_{reject}, \sqcup, R), (q_1, a) \to (q_1, a, R), (q_1, b) \to (q_2, x, L), (q_1, x) \to (q_1, x, R),\\
&\quad (q_2, \sqcup) \to (q_0, \sqcup, R), (q_2, a) \to (q_2, a, L), (q_2, b) \to (q_2, b, L), (q_2, x) \to (q_2, x, L),\\
&\quad (q_3, \sqcup) \to (q_{reject}, \sqcup, R), (q_3, a) \to (q_2, x, L), (q_3, b) \to (q_3, b, R), (q_3, x) \to (q_3, x, R)\}
\end{aligned}
$$

ب) ایده‌ی این بخش نیز کاملاً مشابه حالت قبل است، با این تفاوت که به‌ازای هر مرحله در قسمت قبل که ماشین به دنبال یک b می‌گشت، در این قسمت، باید به دنبال دو عدد b بگردد. این عمل با افزودن دو حالت q_4 و q_5 که قبل از q_1 و q_3 ی حالت قبل عمل می‌کنند انجام می‌شود.

از سویی، بعد از پیدا کردن یک عنصر a در حالت q_0، به جای آن‌که ماشین مستقیماً به q_1 برود، به حالت جدیدی به نام q_4 می‌رود و سپس با مشاهده‌ی اوّلین b، به q_1 می‌رود و فی‌الواقع با این عمل به دنبال دو عنصر b برای هر a می‌گردد. قوانین جدید و تغییرات قبلی چنین است:

$$
\begin{aligned}
&\delta(q_0, a) = (q_4, x, R),\\
&\delta(q_4, \sqcup) = (q_{reject}, \sqcup, R),\ \delta(q_4, a) = (q_4, a, R),\ \delta(q_4, x) = (q_4, x, R),\ \delta(q_4, b) = (q_1, x, R)
\end{aligned}
$$

از سوی دیگر، باید در نظر داشت که با مشاهده‌ی اوّلین عنصر b در حالت q_0، الزاماً باید به دنبال b ی دوم گشته و سپس به q_3 رفت که این عمل با افزودن q_5 و قوانین مربوط به خودش مسجّل می‌گردد:

$$
\begin{aligned}
&\delta(q_0, b) = (q_5, x, R),\\
&\delta(q_5, \sqcup) = (q_{reject}, \sqcup, R),\ \delta(q_5, a) = (q_5, a, R),\ \delta(q_5, x) = (q_5, x, R),\ \delta(q_5, b) = (q_3, x, R)
\end{aligned}
$$

[2] reject
[3] Formal description

مسأله‌ی ۲:

الف) برای این منظور، کافی‌ست یک زبان L ارائه دهیم که توسّط 2-PDA پذیرفته شود ولی توسّط 1-PDA قابل تشخیص و پذیرش نباشد.
این زبان را چنین تعریف می‌کنیم:

$$L = \{w\#w | w \in \{۰, ۱\}^*\}$$

از آن‌جا که (چنان که در کتاب اثبات شده)، این زبان مستقل از متن[۴] نمی‌باشد، توسّط 1-PDA که یک PDA ی معمولی است، قابل پذیرش نیست.
امّا از سوی دیگر این زبان توسّط یک 2-PDA که یک PDA با دو پشته[۵] است، قابل پذیرش است. نحوه‌ی عمل‌کرد 2-PDA نیز این‌گونه است که نیمه‌ی اوّل ورودی (تا جایی که در ورودی # دیده شود) را خوانده و در پشته‌ی اوّل می‌ریزد. سپس همزمان با خواندن # از ورودی، تمامی محتوای پشته‌ی اوّل را در پشته‌ی دوم می‌ریزد. پس از آن، از آن‌جا که دو پشته به صورت $FILO$[۶] هستند و نیمه‌ی اوّل ورودی از طریق پشته‌ی اوّل وارد پشته‌ی دوم شده‌است، ترکیب آن دو به صورت $FIFO$[۷] عمل کرده و اوّلین (بالاترین) عنصر پشته‌ی دوم، اوّلین (سمت چپ ترین) عنصر w خواهد بود. از این رو کافی‌ست برای بررسی این‌که رشته به صورت $w\#w$ است یا نه، مشابه کاری که برای زبان $w\#w^R$ می‌کردیم را انجام دهیم و همزمان با خواندن ورودی، عناصر پشته‌ی دوم را نیز استخراج[۸] کنیم. در صورتی که این دو عنصر برابر نبودند، اعلام عدم پذیرش می‌کنیم. در پایان نیز اگر ورودی و پشته‌ی دوم هم‌زمان تمام شدند (به انتها رسیدند)، اعلام پذیرش می‌کنیم و در غیر این‌صورت اعلام عدم پذیرش.

ب) اگر کلّ سیستم پشته‌ها را به‌صورت یک جعبه‌ی سیاه[۹] در نظر بگیریم، تنها کافی‌ست اعمال $Push$ و Pop را برای هر سه پشته‌ی 3-PDA، با استفاده از دو پشته‌ی 2-PDA شبیه‌سازی کنیم. برای این منظور، اگر محتوای هر سه پشته‌ی 3-PDA را (با قرار دادن یک عنصرِ $\# \notin \Sigma$ قبل از هر رشته و نیز در پایان کلّ رشته‌ها، نظیر $\#C_۱\#C_۲\#C_۳\#$ که C_i محتوای پشته‌ی i اُمِ 3-PDA است) در پشته‌ی اوّلِ 2-PDA بریزیم، در هر مرحله برای دسترسی به پشته‌ی i اُمِ 3-PDA (که $۱ \leq i \leq ۳$) کافی‌ست،

---

[۴] $Context\ Free$
[۵] $Stack$
[۶] $FirstIn, LastOut$
[۷] $FirstIn, FirstOut$
[۸] Pop
[۹] $Black\ Box$

- تا رسیدن به i اُمین #، عناصر پشته‌ی اوّلِ 2-PDA را، با اعمالی نظیر (Push(2, Pop(1))، در پشته‌ی دوم بریزیم. در پایان این مرحله بالاترین عنصرِ پشته‌ی دوم همان i اُمین #ِ کلّی است.
- سپس عمل مورد نظر را با پشته‌ی اوّلِ 2-PDA (که اکنون مشابه پشته‌ی i اُمِ 3-PDA است) انجام دهیم. بدیهی‌ست که اگر در این قسمت به # رسیدیم، معادل این‌ست که در 3-PDA به انتهای پشته‌ی مربوطه رسیده‌ایم.
- پس از آن مجدداً، با اعمالی نظیر (Push(1, Pop(2))، محتوای پشته‌ی دومِ 2-PDA را در پشته‌ی اوّل بریزیم. بدیهی‌ست که پس از این مرحله پشته‌ی دوم خالی بوده و محتوای پشته‌ی اوّل برابر محتوای پشته‌های 3-PDA ی فرضی‌ست که با # به هم چسبیده‌اند.

بدیهی‌ست که چون بخش پشته‌ها مستقل از بخش حالات[۱۰] و به صورت یک جعبه‌ی سیاه بررسی شده، نیازی به درگیر شدن با حالات نیست.

.....

مسأله‌ی ۳:

فرض می‌کنیم که ماشین M_1 یک ماشین تصمیم‌گیرنده برای زبان تصمیم‌پذیر L_1 و ماشین M_2 نیز یک ماشین تصمیم‌گیرنده برای زبان تصمیم‌پذیر L_2 است.

الف) ماشین M_3 را بدین صورت می‌سازیم که این ماشین با دریافت ورودی w، آن را به دو ماشین M_1 و M_2 می‌دهد. از آن‌جا که این دو ماشین تصمیم‌گیرنده هستند، در حلقه[۱۱] نخواهند افتاد و جواب پذیرش[۱۲] یا رد[۱۳] خواهند داد. اکنون ماشین M_3 در صورتی که حداقل یکی از جواب‌های $<M_1, w>$ یا $<M_2, w>$ برابر پذیرش بود جواب پذیرش و در غیر این‌صورت جواب رد می‌دهد.

واضح است که زبانی که ماشین M_3 می‌پذیرد $L_1 \bigcup L_2$ خواهد بود؛ چرا که هر ورودی w که توسّط ماشین M_3 پذیرفته شود، حتماً توسّط یکی از M_1 یا M_2 پذیرفته شده و از این رو عضو یکی از دو زبان L_1 یا L_2 بوده‌است. از سوی دیگر، هر رشته‌ای که توسّط یکی از این دو ماشین پذیرفته شود نیز توسّط M_3 پذیرفته خواهد شد.

ب) ماشین M_4 را بدین صورت می‌سازیم که این ماشین با دریافت ورودی w، مشابه حالت قبل، آن

---

[۱۰] $States$
[۱۱] $loop$
[۱۲] $accept$
[۱۳] $reject$

را به دو ماشین M_1 و M_2 می‌دهد، با این تفاوت که جواب ماشین، $<M_1, w> \wedge <M_2, w>$ [14] خواهد بود. که $<M_1, w>$ و $<M_2, w>$ جواب‌های (بولی) ماشین‌های M_1 و M_2 برای ورودی w خواهند بود.

واضح است که زبانی که ماشین M_4 می‌پذیرد، با اثباتی مشابه حالت قبل، برابر $L_1 \bigcap L_2$ خواهد بود.

ج) ماشین تصمیم‌پذیر M_5 را برای پذیرش زبان $L_1{}^*$ به این ترتیب می‌سازیم:

- در صورتی که رشته‌ی ورودی برابر ϵ بود آن را بپذیر؛
- در غیر این‌صورت به‌صورت غیر قطعی[15]، رشته‌ی w ی ورودی را به دو تکّه‌ی w_1 و w_2 طوری تقسیم کن که $w = w_1 \cdot w_2$. سپس در صورتی که w_2 از تکرار w_1 به دست نمی‌آمد، مقدار مردود را برگردان؛ در غیر این‌صورت، خروجی ماشین M_1 را برای w_1 به عنوان خروجی نهایی برگردان.

بدیهی‌ست که این ماشین، یک ماشین تورینگ غیرقطعی $(N-Turing)$ است که طبق مطالب گفته شده در کتاب، چون یک تصمیم‌گیرنده برای زبان $L_1{}^*$ است، پس این زبان تصمیم‌پذیر است.

تنها نکته‌ی قضیه، نحوه‌ی بررسی این‌که «آیا w_2 از تکرار w_1 به دست می‌آید یا نه؟» است که این مهم نیز به راحتی و با مارک کردن دسته به دسته‌ی عناصر w_2 در صورت مشابهت با عناصر w_1 تا رسیدن به آخرین عنصر و سپس برگرداندن جواب و ...، از طریق یک ماشین تورینگ امکان‌پذیر است.

د) ماشین تصمیم‌پذیر M_6 را برای پذیرش زبان $L_1 \cdot L_2$ به این ترتیب می‌سازیم:

- به صورت غیرقطعی، رشته‌ی w ی ورودی را به دو تکّه‌ی w_1 و w_2 طوری تقسیم کن که $w = w_1 \cdot w_2$.
- مقدار $<M_1, w_1> \wedge <M_2, w_2>$ را برگردان.

بدیهی‌ست که چون ماشین M_6 غیرقطعی‌ست، پس در صورتی که $w \in L_1 \cdot L_2$ باشد، حتماً جواب پذیرش را بر می‌گرداند.

ه) برای ساختن ماشین تصمیم‌پذیر M_7، برای پذیرش زبان $\overline{L_1}$، کافی‌ست ورودی ماشین M_7 را به عنوان ورودی به ماشین M_1 داده و مقدار عکس (not) آن را برگردانیم.

. . . . . .

[14] جواب مقدار بولی «صحیح یا $true$» برای یک ماشین تورینگ همان «پذیرش» و مقدار «غلط یا $false$» به معنای «رد یا عدم پذیرش» است.

[15] $non\ deterministic$

مسأله‌ی ۴:

مشابه سؤال قبل، فرض می‌کنیم که ماشین M_1 یک ماشین تشخیص‌دهنده برای زبان تشخیص‌پذیر L_1 و ماشین M_2 نیز یک ماشین تشخیص‌دهنده برای زبان تشخیص‌پذیر L_2 است.

الف) ماشین M_3 را بدین صورت می‌سازیم که این ماشین با دریافت ورودی w، آن را به دو ماشین M_1 و M_2 می‌دهد. در صورتی که حداقل یکی از جواب‌های $<M_1, w>$ یا $<M_2, w>$ برابر پذیرش بود، این ماشین جواب پذیرش می‌دهد. در صورتی که هر دو جواب رد دادند، جواب رد می‌دهد و در غیر این‌صورت (که یکی در حلقه و دیگری رد، یا هر دو در حلقه باشند)، بدون پاسخ می‌گذارد.

ب) ماشین M_4 را بدین صورت می‌سازیم که این ماشین با دریافت ورودی w، آن را به دو ماشین M_1 و M_2 می‌دهد. در صورتی که هر دو جواب $<M_1, w>$ و $<M_2, w>$ پذیرش بودند، جواب پذیرش می‌دهد. در صورتی که یکی از آن دو جواب برابر رد باشد، جواب رد می‌دهد و در غیر این‌صورت (که هر دو در حلقه باشند یا یکی پذیرش و دیگری در حلقه)، بدون پاسخ می‌گذارد.

ج) ماشین تشخیص‌پذیر M_5 را برای پذیرش زبان $L_1{}^*$ به این ترتیب می‌سازیم:

- در صورتی که رشته‌ی ورودی برابر ϵ بود آن را بپذیر؛
- در غیر این‌صورت به‌صورت غیر قطعی، رشته‌ی w ی ورودی را به دو تکّه‌ی w_1 و w_2 طوری تقسیم کن که $w = w_1 \cdot w_2$. سپس در صورتی که w_2 از تکرار w_1 به دست نمی‌آمد، مقدار مردود را برگردان؛ در غیر این‌صورت، خروجی ماشین M_1 را برای w_1 به عنوان خروجی نهایی برگردان.

بدیهی‌ست که این ماشین، یک ماشین تورینگ غیرقطعی $(N-Turing)$ است که طبق مطالب گفته شده در کتاب، چون یک تشخیص‌دهنده برای زبان $L_1{}^*$ است، پس این زبان تشخیص‌پذیر است.
تنها نکته‌ی قضیه، نحوه‌ی بررسی این‌که «آیا w_2 از تکرار w_1 به دست می‌آید یا نه؟» است که این مهم نیز به راحتی و با مارک کردن دسته به دسته‌ی عناصر w_2 در صورت مشابهت با عناصر w_1 تا رسیدن به آخرین عنصر و سپس برگرداندن جواب و ...، از طریق یک ماشین تورینگ امکان‌پذیر است.

د) ماشین تشخیص‌پذیر M_5 را برای پذیرش زبان $L_1 \cdot L_2$ به این ترتیب می‌سازیم:

- به صورت غیرقطعی، رشته‌ی w ی ورودی را به دو تکّه‌ی w_1 و w_2 طوری تقسیم کن که $w = w_1 \cdot w_2$.
- در صورتی که $<M_1, w_2>$ یا $<M_2, w_2>$ در حلقه افتادند، بدون پاسخ بمان و در غیر این‌صورت، مقدار $<M_1, w_1> \wedge <M_2, w_2>$ را برگردان.

بدیهی‌ست که چون ماشین $M_۶$ غیرقطعی‌ست، پس در صورتی که $w \in L_۱ \cdot L_۲$ باشد، حتماً جواب پذیرش را بر می‌گرداند.

.....

پایان