به نام خدا

نظریه زبان‌ها و ماشین‌ها - بهار ۱۴۰۱

پاسخ تمرین شماره ۱۱

دستیار آموزشی این مجموعه: محمدطاها فخاریان

taha.fakharian@gmail.com

---

۱) حالتی از PCP را در نظر بگیرید که در آن، طول هر رشته بالای هر دومینو با طول هر رشته پایین هر دومینو برابر باشند. برای مثال در نظر بگیرید: $L = \{\frac{ab}{aa}, \frac{a}{b}, \frac{abaa}{bbba}\}$ در این دسته قرار می‌گیرد. نشان دهید این حالت از مسأله PCP تصمیم‌پذیر است.

برای حل این مسأله، به این مورد باید دقت داشته باشید که طول رشته بالا و پایین هر جفت باهم برابر است. حال فرض کنید یک match یافته‌ایم که با $\frac{a}{a'}$ شروع می‌شود، با توجه به اینکه $|a| = |a'|$، پس باید a, a' باهم برابر باشند. به همین ترتیب بر پایه اولین جفت می‌توان برای جفت‌های دیگر نیز با استقرا ثابت کرد که باید رشته‌های بالا و پایینشان باهم برابر باشند. همچنین واضح است که اگر هیچ جفتی با این ویژگی نداشته باشیم، مسأله پاسخ نخواهد داشت. بنابراین برای بررسی اینکه این مسأله پاسخ دارد یا نه، کافیست بررسی کنیم که آیا جفتی وجود دارد که رشته‌های بالا و پایین آن یکسان باشند یا خیر، در صورتی که این جفت وجود داشته باشد، آن را می‌پذیریم و در غیر این صورت رد می‌کنیم(از آنجا که تعداد جفت ورودی‌های ورودی متناهی است، بنابراین مسأله تصمیم‌پذیر است).

۲) اگر G_1, G_2 دو گرامر مستقل از متن باشند، ثابت کنید زبان‌های زیر *Undecidable* هستند(می‌توانید فرض کنید که زبان $ALL_{CFG} = \{< G > | L(G) = \Sigma^*\}$ یک زبان *Undecidable* است).

$a.\ L_1 = \{< G_1, G_2 > | L(G_1) \cap L(G_2) \neq \emptyset\}$
$b.\ L_2 = \{< G_1, G_2 > | L(G_1) = L(G_2)\}$

a.

مسأله را به کمک کاهش PCP و برهان خلف حل می‌کنیم. فرض می‌کنیم که زبان L_1 تصمیم‌پذیر است و ماشین تورینگ آن D است. Decider زبان PCP را به این صورت می‌سازیم:

اگر دومینوها را به صورت $\{\frac{x_1}{y_1}, \cdots, \frac{x_n}{y_n}\}$ در نظر بگیریم، دو گرامر A و B را به صورت زیر می‌سازیم(۱ تا n، اعداد در مبنای n+1 هستند):

$$A \rightarrow x_1A1|x_2A2|\cdots|x_nAn|x_11|\cdots|x_nn$$
$$B \rightarrow y_1B1|y_2B2|\cdots|y_nBn|y_11|\cdots|y_nn$$

حال این دو گرامر را به D می‌دهیم: در صورتی که آن را accept کرد، ورودی را accept می‌کنیم و در غیر این صورت آن را reject می‌کنیم. این یک تصمیم‌گیر برای PCP است، در صورتی که می‌دانیم اینطور نیست. از تناقض بوجود آمده، درستی حکم را نتیجه می‌گیریم.

b.

مسأله را به کمک کاهش All_{CFG} و برهان خلف حل می‌کنیم. فرض می‌کنیم ماشین تورینگ زبان L_2 D باشد. Decider زبان All_{CFG} را به این صورت می‌سازیم. گرامر A را به صورت زیر می‌سازیم($a_1, \cdots, a_n$ همه اعضای الفبای گرامر G هستند):

$$A \rightarrow a_1A|\cdots|a_nA|\epsilon$$

رشته $< A, G >$ را به D می‌دهیم: در صورتی که آن را accept کرد، ورودی را accept می‌کنیم و در غیر این صورت آن را reject می‌کنیم. این یک تصمیم‌گیر برای All_{CFG} است، در صورتی که می‌دانیم اینطور نیست. از تناقض بوجود آمده، درستی حکم را نتیجه می‌گیریم.

۳) در صورتی که الفبای زبان *{0,1}* باشد، ثابت کنید $L, \bar{L}$ هر دو *Turing-recognizable* نیستند.

$$\boldsymbol{A = \{< M, w > | M \text{ is a turing machine that accepts } w\}}$$

$$\boldsymbol{L = \{0x | x \in A\} \cup \{1y | y \notin A\}}$$

از کاهش زبان $\overline{A_{TM}}$ و برهان خلف استفاده می‌کنیم. فرض می‌کنیم *recognizer* زبان L R باشد. *Recognizer* زبان $\overline{A_{TM}}$ را به این صورت می‌سازیم. به ازای ورودی $< M, w >$، عدد یک را به ابتدای w اضافه می‌کنیم و رشته جدید را به همراه تعریف $< M >$ به R می‌دهیم. با توجه به اینکه ابتدای رشته ۱ است،

R در صورتی که رشته w(بدون احتساب ۱ اضافه شده) در زبان L وجود نداشته باشد، آن را $accept$ می‌کند و در غیر این صورت یا $reject$ کرده یا $loop$ می‌زند. این یک $recognizer$ برای $\overline{A_{TM}}$ است، در صورتی که می‌دانیم اینطور نیست. از تناقض بوجود آمده، درستی حکم را نتیجه می‌گیریم. برای $\bar{L}$ نیز به مشابه قبلی عمل می‌کنیم، با این تفاوت که به ابتدای w عدد صفر را اضافه می‌کنیم. با استدلال مشابه قسمت قبل، نتیجه می گیریم که این زبان نیز $recognizable$ نیست.

۴) یک وضعیت بی‌فایده در یک ماشین تورینگ، وضعیتی است که با هیچ رشته ورودی نتوان به آن وارد شد. ثابت کنید مسأله مشخص کردن وجود وضعیت بی‌فایده در یک ماشین تورینگ $Turing\text{-}undecidable$ است.

مسأله را می‌توان به این صورت نوشت:

$$USELESS_{TM} = \{< M, q > | q \text{ is useless in } M\}$$

از کاهش زبان E_{TM} و برهان خلف استفاده می‌کنیم. فرض می‌کنیم که ماشین تورینگ زبان فوق، D باشد. $Decider$ زبان E_{TM} را به این صورت می‌سازیم: برای هر ماشین تورینگ M با وضعیت پذیرنده q_{accept}، این وضعیت بی‌فایده خواهد بود اگر و تنها اگر زبان ماشین تورینگ M تهی باشد. از این نکته استفاده می‌کنیم و تصمیم‌گیر زبان E_{TM} را به این صورت می‌سازیم:

۱- ورودی $< M >$ را بگیر.

۲- ماشین D را روی $< M, q_{accept} >$ اجرا کن که q_{accept} وضعیت پذیرنده در M است.

۳- اگر D پذیرفت، تو هم بپذیر و در غیر این صورت، reject کن.

این یک تصمیم‌گیر برای E_{TM} است، در صورتی که می‌دانیم اینطور نیست. از تناقض بوجود آمده، درستی حکم را نتیجه می‌گیریم.

۵) فرض کنید زبان L شامل زوج‌های $< M, w >$ است، به صورتی که در آن M ماشین تورینگ تک‌نواره‌ای است که بخشی از نوار که ورودی روی آن نوشته شده است را هرگز تغییر نمی‌دهد و w نیز ورودی این ماشین است. آیا L تصمیم‌پذیر است یا نه؟ پاسخ خود را اثبات کنید.

ثابت می‌کنیم که تصمیم‌ناپذیر است. برای این کار از برهان خلف استفاده میکنیم و فرض میکنیم L تصمیم‌پذیر باشد و D نیز *decider* آن باشد. حال با استفاده از این موارد یک *decider* برای A_{TM} می‌سازیم، به طوری که ورودی $< M, x >$ را دریافت کند و آن را T می‌نامیم. حال کافیست ورودی‌های T را به ورودی‌های D تبدیل کنیم. برای این کار ماشین M' را از روی ورودی‌های T تعریف می‌کنیم، به صورتی که به شکل زیر عمل کند:

۱- در انتهای ورودی x ،یک # مینویسد و x را در سمت راست # کپی میکند(در انتهای این مرحله نوار به صورت x#x درمی‌آید).

۲- x سمت راست # را به عنوان ورودی برای M در نظر گرفته و M را روی آن اجرا میکند(در نتیجه در این مرحله x سمت چپ # تغییری نمیکند).

۳- در صورتی که x،M را بپذیرد به سمت چپ # میآید و مقداری را روی آن تغییر می‌دهد(پس در این مرحله x سمت چپ # نیز تغییر میکند).

با توجه به این تغییرات، $< M', x >$ را به عنوان ورودی به D میدهیم و خروجی D را به خروجی T وصل میکنیم به طوری که اگر D بپذیرد، T رد کند و اگر D رد کرد، T بپذیرد(زیرا اگر D ورودی را بپذیرد، یعنی M' ورودی x را تغییر نداده که با توجه به مرحله ی c در تعریف M'، نتیجه میگیریم که M ورودی x را نپذیرفته و اگر D ورودی نپذیرد یعنی مرحله ی c رخ داده و در نتیجه M ورودی خود را پذیرفته است). در نتیجه با توجه به تصمیم ناپذیر بودن A_{TM} ثابت میشود که فرض اشتباه است و تصمیم‌ناپذیر است.

۶) عبارات زیر را اثبات کنید:

الف) زبان A، *Turing-recognizable* است اگر و تنها اگر $A \leq_m A_{TM}$.

ب) زبان A تصمیم‌پذیر است اگر و تنها اگر $A \leq_m 0^*1^*$.

الف) می‌دانیم اگر $A \leq_m A_{TM}$ و B، *recognizable* باشد، A نیز *recognizable* خواهد بود. با توجه به اینکه می‌دانیم A_{TM}، *recognizable* است، طبق توضیحات فوق A نیز *recognizable* خواهد بود.

با توجه به این فرض باید یک تابع کاهش ارائه دهیم که بتوان ثابت کرد $A \leq_m A_{TM}$. با توجه به تعریف A_{TM} داریم(M ماشین تورینگی است که A را توصیف می‌کند):

$$x \in A \rightarrow < M, x > \in A_{TM}$$
$$x \notin A \rightarrow < M, x > \notin A_{TM}$$

بنابراین به عنوان تابع کاهش میتوانیم از تابع $f(x) =< M, x >$ استفاده کنیم. همچنین از آنجا که این تابع محاسبه‌پذیر است پس میتوان از آن برای کاهش استفاده کرد و ثابت میشود که $A \leq_m A_{TM}$.

ب) برای این بخش نیز می‌دانیم اگر $A \leq_m A_{TM}$ و B، *decidable* باشد، A نیز *decidable* خواهد بود. بنابراین با توجه به 0^*1^* تصمیم‌پذیر است، A نیز تصمیم‌پذیر خواهد بود. حال فرض می‌کنیم A تصمیم‌پذیر است و ماشین متناظر با آن، D باشد. تابع زیر را به عنوان تابع کاهش در نظر می‌گیریم:

$$f(x) = \begin{cases} 01 & \mathrm{x} \in A \\ 10 & x \notin A \end{cases}$$

برای این که از این تابع به عنوان تابع کاهش استفاده کنیم باید ثابت کنیم که این تابع قابل محاسبه است. برای انجام این کار کافیست ماشین تورینگی را توصیف کنیم که با توجه به ورودی مقدار $f(x)$ را محاسبه می کند. برای این کار D را روی x اجرا می‌کنیم و ورودی را پاک می‌کنیم: اگر پذیرفت،*01* را می‌نویسیم و در غیر این صورت *10* را می‌نویسیم. طبق این توضیحات، $f(x)$ قابل محاسبه است و لذا تابع کاهش به درستی عمل می‌کند و خواهیم داشت: $A \leq_m 0^*1^*$.

۷)(امتیازی) زبان A شامل $< R, S >$هایی است که در آنها R و S عبارت‌های منظم هستند و $L(R) \subseteq L(S)$. ثابت کنید A زبانی تصمیم‌پذیر است.

می‌دانیم اگر $A \subseteq B$ آنگاه $A \cap \bar{B} = \emptyset$. همچنین میدانیم هر عبارت منظم یک DFA معادل دارد که رشته‌های زبانی را که عبارت منظم مربوطه مشخص می‌کند را می‌پذیرد. علاوه بر می‌دانیم که میتوان از DFA زبان L، DFA زبان $\bar{L}$ را نیز ساخت و با داشتن دو DFA می‌توان اشتراک آن دو را ساخت. حال یک ماشین تورینگ تعریف میکنیم که به ازای دریافت ورودی x به این صورت عمل می‌کند:

۱- ابتدا R و S را از x جدا می‌کند(در صورتی که x به این فرمت نبود، آن را رد می‌کند).

۲- با توجه به عبارات منظم، DFAهای متناظر را ساخته و سپس از روی آنها، $L(R) \cap L(\bar{s})$ را می‌سازد.

۳- با توجه به این که میدانیم بررسی تهی بودن یک زبان منظم تصمیم‌پذیر است، DFA تولید شده را به ماشین تورینگ تشخیص دهنده تهی بودن زبان منظم میدهیم، در صورتی که زبان تهی باشد ورودی را میپذیریم و در غیر این صورت رد میکنیم.