علیرضا کرمی HW10 810101492

۱) برای اینکه دو زبان $L(F(M_1))$ و $L(F(M_2))$ نابرابر باشند، باید رشته‌ای وجود داشته باشد که توسط یکی از آن‌ها accept و توسط دیگری reject شود. از آنجایی که رشته‌ها طول مشخص دارند و از آنجا که باید $2^{|w|}$ گام، آن را روی هر یک از ماشین‌ها اجرا کنیم که تعداد گام محدود است، پس قطعا بعد از تعداد گام متناهی، متوقف می‌شود و یا accept یا reject می‌کند (و در loop نمی‌افتد). این برای یک رشته بود. حال برویم سر حل مسئله. در صورتی که دو زبان برابر نباشند، قطعا رشته‌ای مانند w وجود دارد که در یکی از آن‌ها accept و در دیگری reject می‌شود (در یک زبان پایان‌پذیر). این یعنی اگر ماشینی بسازیم که همه رشته‌ها را بخواهیم چک کند، قطعا روزی به رشته w خواهیم رسید (چون مطمئن هستیم چنین رشته‌ای وجود دارد) و نابرابری دو زبان ثابت می‌شود و در نتیجه ما TM مربوط به L را accept می‌کنیم. در صورتی که این دو زبان یکی باشند تا ابد دنبال w می‌گردد و چک می‌کند که برای یکی accept و برای دیگری reject باشد و چون چنین w را پیدا نمی‌کند سراغ w بعدی می‌رود و این کار را تا بی‌نهایت انجام می‌دهد. بنابراین پایان‌پذیر نخواهد بود و در loop خواهد افتاد پس recognizable می‌شود. (برای accept، چون می‌دانیم w مشخص وجود دارد و ما در $2^{|w|}$ step قطعا به آن خواهیم رسید و حتی آن را می‌بینیم، پایان‌پذیر خواهد بود) (خود چک کردن w برای هر دو زبان هم عمل پایان‌پذیر است) ← L مربوط به TM تشخیص‌پذیر است. (دقت کنید تعداد گام‌ها بی‌نهایت نیست و $2^{|w|}$ و عددی مشخص و متناهی است). پس یا accept دادیم یا در loop برای چک کردن می‌افتیم. برای هر کلمه‌ی

۲) می‌دانیم که n عددی مشخص است پس تعداد DFA‌های ساخته شده با کمتر از یا مساوی n^2 است، عددی مشخص و محدود و متناهی است. حال برای تک‌تک این DFA‌ها با کمتر از n است، چک می‌کنیم که $L(DFA) \neq L(F(M))$ باشد (مثل همان قسمت الف). در صورتی که برای تمامی DFA‌ها داشته باشیم $L(DFA) \neq L(F(M))$، یعنی هیچ DFA با کمتر یا مساوی n استیت وجود ندارد که $F(M)$ را توصیف کند و ما accept می‌کنیم. در غیر این صورت، یکی از DFA‌ها می‌تواند $F(M)$ را توصیف کند و یعنی $L(DFA) = L(F(M))$. این اتفاق یعنی برابری این دو زبان و reject می‌شود (چون نابرابر نیستند) ممکن است ثابت شود یا برای چک کردن همانند قسمت الف در loop بیفتد پس recognizable می‌شود. پس اگر چک کردن یکی از آن‌ها برابر شد و reject شد، ما نیز reject می‌کنیم (TM مربوط به L صدا زده شده را)

ادامه 2) در واقع ما دیدیم که $L(DFA) \neq L(F(M))$ زیرمجموعه‌ای از $L(F(M_1)) \neq L(F(M_2))$ است و یعنی تشخیص پذیر است، و برای اینکه برای تمام DFA ها این رابطه برقرار باشد باید اشتراک بگیریم و یعنی داریم: $\underbrace{[L(DFA_1) \neq L(F(M))]}_{\text{recognizable}} \cap \underbrace{[L(DFA_2) \neq L(F(M))]}_{\text{recognizable}} \cap \dots \cap \underbrace{[L(DFA_n) \neq L(F(M))]}_{\text{recognizable}}$

از طرفی طبق اثبات در سوال (3)، ما داریم T-recognizable ها تحت عمل اشتراک بسته هستند. پس TM حاصل که همان زبان L صورت مسئله را توصیف می‌کند نیز recognizable است.

چک کردن هر DFA با F(M) را به صورت نوارهای همزمان پیش می‌بریم که اینکار را می‌توانیم مثلا با یک Multi_tape انجام دهیم و در هر tape یکی از DFA ها را با F(M) چک کنیم:

۳)

برای اجتماع:

یک U_{TM} جدیدی می‌سازیم (یونیورسال TM) که دو تا TM را به صورت موازی روی ورودی اجرا می‌کنیم (حالا هم می‌توانیم به صورت Multi-tape TM یا به صورت Non deterministic این کار را انجام دهیم.) اگر هرکدام از آن‌ها به accept ختم شد، accept می‌کند و در غیر اینصورت reject می‌کنیم. (دلیل موازی کردن این بود که اگر سری می‌کردیم ممکن بود یکی به loop بیفتیم و accept بعدی را نبینیم هیچ وقت.)

برای اشتراک:

مثل قبل یک TM درست می‌کنیم که در آن، دو ماشین تورینگ A و B را به صورت موازی روی ورودی آن اجرا می‌کنیم. در صورتی که هر دو، به accept ختم شد، رشته را accept کرده و در غیر این صورت reject می‌کنیم.

۴) می‌دانیم که DFA ها تحت عمل reverse کردن، بسته هستند (کافی است که استیت‌های accept را یک استیت initial state کرده و آن را تبدیل به initial state می‌کنیم و همچنین تمام transition ها را برعکس می‌کنیم یعنی $\delta(q_1, a) = q$ تبدیل می‌شود به $\delta(q_2, a) = q_1$).

حال که DFA برعکس شده را ساختیم، آن را به همراه DFA اصلی به ماشین تورینگ EQ_{DFA} می‌دهیم تا چک کند که دو DFA معادل هم هستند یا خیر. از آنجایی که EQ_{DFA} تصمیم‌پذیر است، پس زبان L نیز تصمیم‌پذیر خواهد بود. اگر EQ_{DFA} accept کند، یعنی هم خود رشته و هم reverse آن را پذیرفته است و در زبان ما هستند و L نیز accept می‌کند. در غیر اینصورت، L نیز reject خواهد کرد.

5) فرض کنید که رشته w را داشته باشیم. اینکه رشته w یک Palindrome است را می‌توان به کمک یک CFG تشخیص داد که در تمارین قبل حل کردیم.

$$S \to XSX \mid X \mid \varepsilon \quad x \in \Sigma$$

پس اینکه w یک رشته تصمیم‌پذیر است یا CFG قابل بیان است و تصمیم‌پذیر است. حال تمام حالت‌های انشعاب w به دو رشته w_1 و w_2 را در نظر می‌گیریم. (مثلا داریم:)

$w = 01011101$

$w_1 = \emptyset$, $w_2 = 01011101$

$w_1 = 0$, $w_2 = 1011101$

$w_1 = 01$, $w_2 = 011101$

$w_1 = 010$, $w_2 = 11101$

$\vdots$

$w_1 = 01011101$, $w_2 = \emptyset$

حال اگر یکی از حالت‌های $w_1 \in L_1$ بود باشد و $w_2 \in L_2$ (هر دو تصمیم‌پذیرند) و همچنین w به صورت Palindrome باشد (که آن هم تصمیم‌پذیر است)، اشتراک چند زبان Turing_decidable، تصمیم‌پذیر خواهد بود.

(کافی است رشته ورودی را روی هر کدام از زبان‌ها تصمیم‌پذیر اجرا کنیم و در صورتی که توسط همهٔ ماشین‌ها پذیرفته شد، رشته ورودی را accept می‌کنیم و در غیر اینصورت reject می‌کنیم و می‌دانیم که چون هر کدام از ماشین‌ها تصمیم‌پذیر است، پایان‌پذیر خواهد بود و همهٔ ماشین‌ها در کنار هم پایان‌پذیر خواهند بود و loop نداریم. پس decidable خواهد بود) در واقع زبان‌های Turing_decidable تحت عمل اشتراک بسته هستند

پس اگر هر سه accept کردند، ما نیز accept می‌کنیم و در غیر اینصورت reject می‌کنیم.

حال اگر تنها یکی از این انشعاب‌ها به accept ختم نشود، L accept می‌کند و اگر هیچ‌کدام accept نشد برای زبان L reject می‌شود. در واقع اجتماع گرفتیم و زبان‌های Turing decidable تحت عمل اجتماع بسته هستند

(مثل قبل نظر که برای اشتراک گفتیم، فقط اینجا اگر یکی هم accept شد accept می‌کنیم و در غیر اینصورت reject)

در واقع:

$$L = (A_1 \cap B_1 \cap C_1) \cup (A_2 \cap B_2 \cap C_2) \cup \dots \cup (A_n \cap B_n \cap C_n)$$

$A \to w_1 \in L_1$

$B \to w_2 \in L_2$

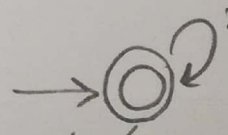
$C \to w \in \text{palindrom}$

$A_i \cap B_i \cap C_i \to$ برای اشتراق i ام

$|w_1| = i$, $|w_2| = n - i$

۴) از آنجایی که ما تنها به دنبال این هستیم که بررسی کنیم که CFG ما، رشته ای به طول ۰ را می سازد یا خیر پس ماهیت رشته اهمیتی ندارد (اینکه از چه حرف ها و ترمینال هایی تشکیل شده است) و تنها طول رشته اهمیت دارد. پس می توانیم به جای استفاده از ترمینال های مختلف، تنها یک ترمینال قرار دهیم. یعنی در نهایت، رشته تولید شده توسط CFG، به جای داشتن فرمی به صورت $x_1 x_2 \dots x_n$ که x_i ها ترمینال هستند، فرمی به صورت $xx\dots x$ خواهد داشت (تمام ترمینال ها را به یک حرف map کردیم) در واقع CFG خود را با داشتن چندین ترمینال، با اعمال تابعی، تبدیل یا reduce کردیم به CFG با یک ترمینال و اگر CFG با یک ترمینال، تصمیم پذیر باشد، CFG اصلی با چند ترمینال نیز تصمیم پذیر خواهد بود. و آن را M می نامیم

با توجه به قضیه مطرح شده، CFG با یک حرف متکلم است و می توان DFA آن را ساخت. از طرفی می دانیم برای اینکه با کمک یک حرف، رشته ای با هر طول تولید کنیم، می توانیم آن را به صورت زیر نمایش دهیم:



$\Rightarrow$ D: زبان DFA با یک حرف با طول رشته دلخواه

حال DFA ساخته شده M را چک می کنیم که با D برابر است یا نه (از EQ_{DFA} استفاده می کنیم که decidable است) درصورتی که برابر بودند، یعنی رشته با هر طول را تولید می کند و accept می کنیم و در غیر اینصورت reject.

حال چون از reducibility استفاده کردیم، زبان L (CFG با چند ترمینال) نیز زبانی که باعث accept، EQ_{DFA} شد، accept می کند و در غیر اینصورت reject می کند. مراحل به طور خلاصه:

1) از روی L (CFG با چند ترمینال)، زبان L' (CFG با یک ترمینال را می سازیم) (با reducibility) با تورینگ ماشین های k برای L و k' برای L' ($k \leq k'$) اگر k' decidable بود $\leftarrow$ decidableck است.

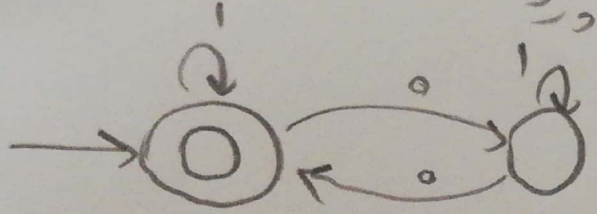
2) برای L'، DFA طراحی می کنیم

3) EQ_{DFA} روی آن DFA و D (DFA با لوپ روی یک ایت accept) اجرا می کنیم.

4) اگر EQ_{DFA}، accept شد $\leftarrow$ k' accept می کند و در غیر اینصورت $\leftarrow$ k' reject می کند.

5) به همین شکل برای k

7) یک DFA به نام D' طراحی می‌کنیم که کلمات با تعداد زوج صفر را بپذیرد.



DFA صورت سوال را با D نشان می‌دهیم.

اگر $L(D) \cap L(D') = \emptyset$ بود یعنی DFA ما، هیچ رشته با تعداد زوج صفر ندارد. در این صورت ماشین تورینگ برای DFA ما، accept می‌کند. اگر $L(D) \cap L(D') \neq \emptyset$ بود، TM ما reject می‌کند (در واقع می‌دانیم که دو DFA، یک عملیات اشتراک بسته هستند، پس DFA حاصل از D و D' را به تورینگ ماشین E_{DFA} می‌دهیم تا بررسی کند که زبان DFA حاصل، empty است یا خیر. در درس یاد گرفتیم که E_{DFA}، decidable است پس TM ما با کمپت E_{DFA}، اکسپت می‌کند و بالعکس). در واقع با اعمال تغییراتی آن را به حل E_{DFA}، reduce کردیم.