نظریه زبان‌ها و ماشین‌ها

دانشگاه فردوسی مشهد

پروژه دوم

نیمسال دوم تحصیلی ۱۴۰۱-۱۴۰۰

گروه مهندسی کامپیوتر | مهلت ارسال: ۱۸ خرداد ساعت ۲۳:۵۹

فاز اول: پیاده‌سازی شبیه‌ساز FST غیرقطعی

در فاز اول این پروژه قصد داریم شبیه‌ساز یک non-deterministic finite state transducer را پیاده‌سازی کنیم.

FST چیست؟

عمده ماشین‌هایی که در درس با آن‌ها آشنا شده‌اید، accepter بوده‌اند. در این دسته از ماشین‌ها، تنها یک نوار ورودی وجود داشت و خروجی به صورت accept یا reject شدن رشته ورودی بود.

FST (Finite state transducer) ماشینی با یک نوار ورودی و یک نوار خروجی است که قابلیت تولید یک رشته در نوار خروجی را دارد. این ماشین تعداد محدودی حالت دارد که هرکدام با یک Transition به حالت دیگری متصل است. هر Transition یک برچسب input:output دارد. (به این معنا که با انجام این Transition مقدار input از رشته ورودی خوانده می‌شود و مقدار output در نوار خروجی نوشته می‌شود)

تفاوت ماشین FST با FSA (که در درس با انواع آن آشنا شدیم) در نحوه تولید خروجی است. ماشین FST با انجام یک Transition علاوه بر اینکه به حالت مقصد می‌رود، در نوار خروجی هم مقداری تولید می‌کند.

همانطور که در درس خواندید یک NFA با پنج‌گانه $(Q, \sum, q_0, F, \delta)$ = M به صورت زیر تعریف می‌شود:

Q: a finite set of states

$\sum$: input alphabet

$q_0 \in Q$: starting state

$F \subset Q$: set of final state

$\delta : Q \times \sum \cup \{\lambda\} \to P(Q)$: transition function between states; given a state $q \in Q$ and a string $w \in \sum^*$, $\delta(q, w)$ returns a set of states $P(Q) \subset Q$.

یک Non-determinisitc FST با شش‌گانه $M = (Q, \sum, \Gamma, q_0, F, \delta)$ تعریف می‌شود:

Q: a finite set of states

$\sum$: input alphabet

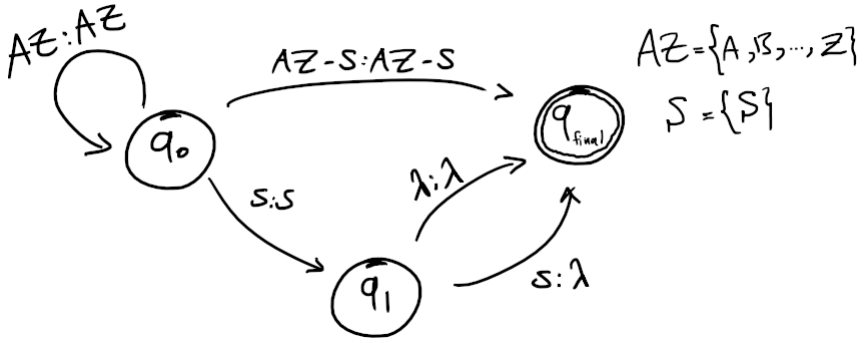
Γ: output alphabet

$q_0 \in Q$: starting state

$F \subset Q$: set of final state

$\delta : Q \times \sum \cup \{\lambda\} \times \Gamma \cup \{\lambda\} \rightarrow P(Q)$: transition relation; given a state $q \in Q$ and a string $w \in \sum^*$, $\delta(q, w)$ generates the output $z \in \Gamma^*$ and returns a set of states $P(Q) \subset Q$.

در یک Non-determinisitc FST اگر رشته ورودی بتواند به یکی از حالات پایانی ختم شود، ماشین رشته را accept کرده و رشته (یا رشته‌های [1]) خروجی را در نوار خروجی تولید می‌کند، در غیر اینصورت ماشین رشته را reject کرده و در نوار خروجی چیزی تولید نمی‌کند. به عنوان مثال ماشین زیر را درنظر بگیرید:



این ماشین تمام رشته‌هایی که از حروف انگلیسی تشکیل شده‌اند را accept می‌کند و اگر این رشته به دو حرف ss ختم شود، یکی از دو s اخر را حذف کرده و به عنوان خروجی برمی‌گرداند، در صورتی که رشته ورودی به دو حرف ss ختم نشود، رشته را بدون تغییر، خروجی می‌دهد. همچنین اگر رشته‌ای که شامل کاراکتری خارج از حروف انگلیسی باشد به ماشین داده شود، رشته را reject می‌کند.

به عنوان مثال:

chess $\rightarrow$ ches bus $\rightarrow$ bus

[1] ممکن است یک رشته ورودی چند مسیر مختلف برای رسیدن به حالات پایانی داشته باشد، در این صورت ماشین برای هرکدام از این مسیرها رشته خروجی تولید میکند و چندین خروجی خواهیم داشت

چه چیزی باید پیاده‌سازی شود؟

در گام اول برنامه شما باید به منظور دریافت حالات و Transition های ماشین، توابعی مشابه با توابع زیر داشته باشد:

$add\_state(state\_name, is\_final)$

حالتی با نام state_name را به ماشین اضافه می‌کند؛ is_final نشان می‌دهد این حالت پایانی است یا خیر.

$add\_transition(in\_state\_name, input, output, out\_state\_name)$

این تابع یک Transition از حالتی به نام in_state_name به out_state_name با برچسب input:output برقرار می‌کند. FST با خواندن input و نوشتن output در نوار خروجی، این Transition را انجام می‌دهد. دقت کنید که مقدار input و output در هر Transition ، دقیقا برابر با یکی از الفبای ورودی و خروجی و یا λ است.

$add\_set\_transition(in\_state\_name, input\_set, out\_state\_name)$

این تابع به ازای هر کاراکتر در input_set ، یک Transition از in_state_name به out_state_name برقرار می‌کند، بطوریکه مقدار input و output یکسان است (FST همان کاراکتری را که از نوار ورودی میخواند، در نوار خروجی می‌نویسد)

در گام بعدی برنامه شما باید با استفاده از حالات و Transition هایی که دریافت کرده است، ماشین غیرقطعی FST متناظر را درست کند. برای این کار برای حالت ماشین (State) ، کلاسی درست کنید که در آن Transitions مربوط به آن حالت نگه داری می‌شود.

در نهایت برنامه شما باید تابعی مشابه با $parse\_input(input\_string)$ داشته باشد. بعد از گرفتن حالات و Transition ها در مراحل قبل، این تابع با گرفتن یک رشته ورودی، اگر رشته توسط ماشین accept شود، خروجی (یا خروجی‌هایی) که FST تولید می‌کند را برمی‌گرداند؛ اگر رشته reject شود، رشته FAIL را برمی‌گرداند.

فاز دوم: Morphological Generation

در این فاز قصد داریم یک ماشین FST برای تبدیل اسامی انگلیسی (nouns) به جمع آن‌ها (plural) طراحی کنیم. اکثر اسامی انگلیسی به صورت **noun + s** جمع بسته می‌شوند.

car → cars　　house → houses

اما چند قانون و استثنا وجود دارد:

- اگر اسم مفرد به **-s, -ss, -sh, -ch, -x, -z** ختم شود، برای جمع بستن به اخر آن **-es** اضافه می‌شود.

bus → buses　　marsh → marshes　　lunch → lunches　　tax → taxes
blitz → blitzes　　truss → trusses

- اگر اسم مفرد به **-y** ختم شود و حرف قبل از **-y** بی‌صدا باشد، آخر اسم به **-ies** تبدیل می‌شود.

city → cities　　puppy → puppies

- اگر اسم مفرد به **-y** ختم شود و حرف قبل از **-y** صدادار باشد، فقط **-s** به اخر کلمه اضافه می‌شود.

boy → boys　　ray → rays

- اگر اسم مفرد به **-f** یا **-fe** ختم شود، **-f** یا **-fe** به **-ve** تبدیل می‌شود و **-s** به اخر اسم اضافه می‌شود.

wolf → wolves　　wife → wives

- اگر اخر اسم به **-o** ختم شود، به اخر اسم **-es** اضافه می‌شود.

potato → potatoes　　tomato → tomatoes

چه چیزی باید پیاده‌سازی کنید؟

در گام اول این فاز، یک FST طراحی کنید که یک رشته متشکل از حروف انگلیسی (lower case) به عنوان ورودی می‌گیرد، آن را accept می‌کند و جمع (plural) آن را در نوار خروجی تولید می‌کند.

در گام بعدی ماشینی که طراحی کرده‌اید را با استفاده از توابع $\text{add\_state}(...)$ ، $\text{add\_transition}(...)$، $\text{add\_set\_transition}(...)$ به شبیه‌ساز FST که در فاز اول پیاده کرده‌اید بدهید. سپس هر رشته از فایل test.txt را خوانده و به تابع $\text{parse\_input}(...)$ بدهید و خروجی را نمایش دهید. فایل test.txt حاوی چندین رشته برای تست کردن ماشین طراحی شده در این فاز است.

در طراحی ماشین حتما به نکات زیر توجه کنید:

- ماشینی که طراحی می‌کنید می‌تواند non-deterministic باشد.
- گراف ماشینی که طراحی می‌کنید را در نظر بگیرید؛ به ازای هر رشته ورودی قابل قبول (رشته‌ای که توسط ماشین accept می‌شود)، باید تنها یک مسیر در گراف ماشین از حالت شروع (initial state) به حالت پایانی (final state) وجود داشته باشد. به عبارتی، نباید بتوان با یک رشته ورودی، از دو مسیر مختلف در گراف ماشین به حالت پایانی رسید.
- ماشینی که قرار است طراحی کنید، وظیفه اینکه تشخیص دهد رشته ورودی یک noun باشد را ندارد و فقط صرفا با توجه به قوانین ذکر شده برای جمع بستن، جمع رشته ورودی را تولید می‌کند. به عنوان مثال اگر رشته abcdy به ماشین داده شود باید خروجی abcdies را تولید کند.
- در طراحی ماشین فقط قوانین ذکر شده در صورت پروژه را درنظر بگیرید و استثنائات قوانین را نادیده بگیرید، به عنوان مثال در قوانین ۴ و ۵ استثنائاتی وجود دارد: belief – beliefs, photo – photos

توضیحات تکمیلی پروژه:

- انجام پروژه به صورت انفرادی است.
- برنامه شما حتما باید به صورت شئ‌گرا نوشته شده باشد.
- برای نوشتن برنامه می‌توانید از زبان‌های Python, Java, CPP استفاده کنید.
- در هنگام تحویل پروژه، باید به کد خود تسلط کافی داشته باشید، همچنین درصورت انجام فاز دوم، باید بتوانید خروجی این فاز برای تمام رشته‌های فایل test.txt را نمایش دهید.
- تمام کدهایی که زده‌اید به همراه تصویری از گراف ماشینی که در فاز دوم طراحی کرده‌اید را در قالب فایل zip با نام StudentName_StudentNumber_PRJ۲ ارسال کنید.