گزارش پروژه اول نظریه زبان و ماشین

شرح پروژه طبق pdf واضح است.

مرحله ابتدایی پروژه ← تفکیک ورودی ها از فایل Input.txt

1- الفبا ، 2- متغیرها ، 3- نود شروع ، 4- قوانین ، 5- دستور

باید از Input.txt خوانده شود. ی مورد اول مربوط به یک گرامر است پس در فایل grammer.py یک شی از گرامر با ی ویژگی اول ساخته و هر بار که یک گرامر جدیدی داشته باشیم ، ی مقدار آن داخل ی متغیر مشخص شده ذخیره می شود و یک شی جدید داخل لیست grammer = [] ذخیره می شود

همچنین متغیر operation مقدار گرفته و در مراحل بعد استفاده می شود.

مرحله دوم پروژه ← گرامر ذخیره شده باید به NFA تبدیل شود.

الگوریتم تبدیل گرامر به NFA (ماشین حالت غیرقطعی)

outputs ← states ، transitions ، start ، final_state

این ها شی از class ی NFA هستند ← alphabets ،

باید یک مجموعه از انتقالات داخل دیکشنری ثبت شود که حالت آغاز، پایان و الفبایی که توسط آن از آغاز به پایان می رویم انجام شده، مشخص شود.

ε-transition انتقالی است که در ماشین های غیرقطعی وجود دارد و اثر داشتن با کلیم $A \xrightarrow{\varepsilon} B$ یعنی A بدون هیچ دستوری می توان راحت به B برود. ε در اصل ∅ است

تابع grammer to nfa ← تبدیل گرامر به nfa

یک کی nfa باید ساخته شود

start ← حالت شروع ماشین با شروع گرامر یکیه

states ← برابر با مجموعه variables در grammer است

alphabet ← یکسان است با grammer

rules ← پردازش قسمت کلیدی است

گسترش از چپ یا گسترش از راست هم است و برای یک گرامر، هر دو کی از راست و چپ مطلح است.

1- اگر سمت راست rules ε باشد ← پس از سمت چپ به حالت پایانی NFA انتقال اضافه می‌شود و برعکس

2- اگر آوزن نماد سمت راست قاعده در الفبای گرامر باشد، یک حالت جدید به نام "F" به مجموعه حالات اضافه می‌شود و یک انتقال از چپ به این حالت جدید اضافه می‌شود و سپس این حالت به مجموعه حالات پایانی NFA اضافه می‌شود و برعکس

3- اگر آوزن نماد در سمت راست یک متغیر باشد که نماد غیرنهایی است، یک انتقال از حالت left به اولین نماد در رشته تولیدی انجام می‌شود. سپس اگر در رشته تولیدی یک alphabet یا همان symbol ورودی وجود داشت یک انتقال به حالت بعدی براساس آن symbol ثبت می‌کنیم و برعکس.

مرحله سوم ← تبدیل NFA به DFA

یک شئ DFA ی سازیم به طوری که الفبا و شروع مانند NFA است و یک set از state ها وضعیت ها را با state-num شماره گذاری می کنیم.

تابع NFA to DFA ← تبدیل ماشین غیر قطعی به قطعی

1- یک شئ از NFA داریم

2- state_name ، alphabet ، start در NFA به همین متغیرها داخل کلاس شئ DFA اختصاص داده می شوند

3- یک صف نیاز داریم برای پردازش حالت هایی که نیاز به بررسی دارند.

4- مجموعه حالت های NFA به ترتیب داخل صف برده می شوند و در هر بار اجرای حلقه، یک مجموعه حالت از صف برداشته شده و باید تمام انتقال ها از مبدا مورد نظر به مقصد مورد نظر بررسی شود و مجموعه symbol هایی که با آن از مبدا به مقصد رفتیم در یک مجموعه ذخیره شود.

5- هر انتقالی که پیدا شد، مجموعه آنها در یک set ذخیره می شود و

بررسی می شود که اگر قبلاً این حالات تعریف نشده اند، پس در دیکشنری state_names با نام جدید ذخیره می شود.

این مرحله را آنقدر ادامه می دهیم که دیگر حالت جدیدی نداشته باشیم.

* اگر برای هر حالت و ورودی انتقالی وجود نداشته باشد، یک وضعیت "N" به مجموعه وضعیت ها اضافه می شود

* وضعیت نهایی DFA براساس وضعیت نهایی NFA تایین می شود.

توضیح متمم ← ۱- تمام حالات غیرنهایی به نهایی، و نهایی ها به غیر نهایی تبدیل می شوند

۲- مجموعه حالت های نهایی را از کل کم کرده و سپس آن را به عنوان مجموعه نهایی DFA ارائه می دهد

توضیح اجتماع ← حالت شروع ترکیبی از start های دو تا DFA ورودی است ← پس DFA جدیدی ساخته می شود

* برای هر نماد ورودی بررسی می شود که چه انتقالاتی در هر DFA

وجود دارد و آنها را در DFA نهایی ی ریزیم ← اگر یکی از حالت ها، حالت نهایی در DFA های آغازین باشد، آن را هم داخل DFA نهایی ی ریزیم.

توضیح اشتراک ← یک DFA جدید تولید شده وقتی ی شود برابره با یکی از DFA های ورودی مثلاً DFA1

* شروع DFA جدید ترکیبی از دوتا DFA ورودی است که start داخل q0 ی باشد
* یک صف برای ذخیره حالات ترکیبی ساخته می شود که برای پیگیری حالاتی است که قبلاً بررسی شده است
* یک دیکشنری به نام state_mapping برای نگه داری نام حالات داریم
* حلقه ای داریم که انتقالات دو DFA ورودی را ها کسب و ترکیب ی کند و جواب را در صورت عدم وجود در انتقالات اولیه، به مجموعه انتقالات اضافه اش ی کند.
* حالات را بررسی ی کنیم ← اگر هر دو حالت ترکیب مثلاً (q_1 و q_2) نهایی باشند، حالت ترکیبی در DFA به عنوان حالت نهایی علامت گذاری ی شود.