«به نام خدا»

زهرا بخشنده

سوال 1:

باید با گرفتن یک شیء گرامر و یک رشته، چک کنیم رشته توسط گرامر تولید می شود یا خیر.

برای این کار ابتدا گرامر CFG را به فرم CNF تبدیل میکنیم:

1 – حذف قوانین lambda

2- حذف قوانین یکه (unit productions)

3- حذف متغیر های useless

4- تبدیل قوانین باقیمانده با طول شبه جمله سمت راست بیشتر از 2 به فرم نرمال چامسکی.

که برای اینکار ابتدا یک شی از کلاس Grammar می سازیم و به ترتیب step های بالا را برای آن انجام میدهیم.

پس از تبدیل گرامر به فرم نرمال چامسکی، از الگوریتم cyk برای تشخیص پذیرفته شدن رشته در گرامر استفاده میکنیم که با استفاده از dynamic programming پیاده سازی میشود:

ابتدا برای هر حرف یک جمله ورودی، تمام متغیرهایی که منجر به تولید آن حرف می‌شوند را در مجموعه ای مجزا قرار می‌دهد. بعد همین کار را برای زیرجمله‌های دوحرفی تکرار می‌کند و بعد زیرجمله‌های سه حرفی تا به کل جمله برسد. در نهایت اگر متغیر ابتدایی را در مجموعه متغیرهای جمله پایانی یافت نتیجه می‌گیرد که جمله توسط گرامر تولید شده‌است.

توابع معادل برای هر step در کلاس Grammar پیاده سازی شده است.

سوال 2:

هر ماشین تورینگ می تواند با رشته های binary کدگذاری شود.

حروف، state ها و ... به صورت زیر کدگذاری می شوند:

Encoding Symbols: a : 1 , b : 11 , c : 111 , d : 1111

State Encoding: q1 : 1, q2 : 11, q3 : 111, q4 : 1111

Head Move Encoding: L : 1 , R : 11

Transition Encoding: δ(q1, a) = (q2, b, L) : 10101101101

سپس با کنار هم قرار گرفتن کد transition ها با 00 به عنوان جدا کننده، کل ماشین کد می شود.

به صورت برعکس نیز میتوانیم با داشتن رشته binary یک ماشین تورینگ، آن ماشین تورینگ را decode کنیم.

و با نوشتن کلاس turing_machine و قرار دادن transition های آن در دیکشنری، به ازای هر رشته چک کنیم که رشته توسط transition های آن می تواند تولید شود یا خیر.

سوال 3:

از pda به دلیل داشتن ساختار داده stack، می توان به عنوان یک calculator استفاده کرد.

برای بدست آورن جواب یک expression ابتدا چک می کنیم که قرار گیری پرانتز های آن، به صورت زبان گرامر زیر باشد:

S => (S) | SS | #

سپس ابتدا با استفاده از pda و stack آن عبارت را به صورت postfix می نویسیم. پس از آن می توانیم به راحتی با استفاده از pda با دیدن operator ها، اعداد را از استک، pop کرده و operand را روی آن ها اعمال کنیم.

در این میان اگر به عبارت نامعقولی رسیدیم که در دامنه توابعمان صدق نمی کرد یا به عبارت بی نهایت رسیدیم، یا هر حالت خاصی در عبارتمان دیدیم که جواب نداشت و در فرم درستی نبود، جواب INVALID را در خروجی می دهیم.

در غیر این صورت، در نهایت، جواب نهایی آن expression بدست می آید.

متد های معادل در کلاس PDA و کلاس PDAStack پیاده سازی شده اند. اگر هرجا به عبارت نامعقولی برسیم توابع کلاس، throw new exception میکنند و در انتها INVALID چاپ می شود.