프로그래밍 언어론

Assignment #3

전기컴퓨터공학부 정보컴퓨터공학전공

201524582 정희석

1. Show that the language generated by the following grammar is a regular language:
(아래의 문법으로 생성되는 언어가 정규언어임을 보여라)

S → aSa | a

A. 먼저 주어진 언어는 정규 문법(Type3)이 아니라 CFG(Type2)이다. 정규문법은 R-L, L-L (Right-Linear, Left-Linear)로 이뤄져 있는데 위의 문법은 Non-Terminal 이 가운데에 있기 때문이다.

위의 문법으로 생성되는 언어는

S -> a, aaa, aaaaa, ... a^2n-1 로

L = $\{a^{2n-1} | n > 0\}$이다. 이 언어가 정규언어인지 확인해보면

이 언어가 정규문법에 의해서 생성이 가능하면 이 언어는 정규언어이다. 이 언어를 생성하는 정규문법을 찾아보면

S → aA | a

A → aaA | aa 로 정규문법이 존재한다.

따라서 위의 언어는 정규언어이다.

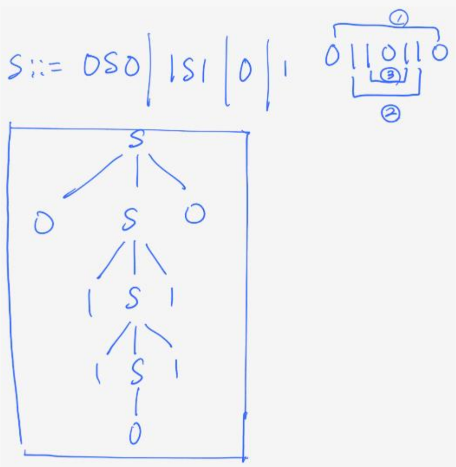
2. Given the context-free grammar

S ::= 0S0 | 1S1 | 0 | 1

Give the derivation tree for 0110110.

(주어진 CFG 에서 0110110 의 derivation tree 을 제시하라)

A.



3. We know that strings of the form anbn require a context-free grammar for their generation. Consider the following regular grammar:
(anbn 은 본래 CFG 를 필요로 한다.)

S ::= aS | bS | a | b

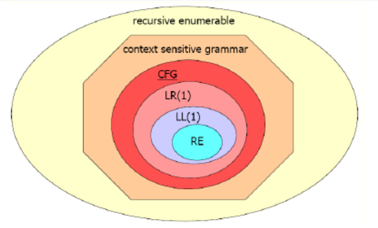
A claim is made that it can generate anbn.
(위의 정규 문법은 anbn 을 나타낸다)

For example, a3b3 is generated by the following derivation:

S => aS => aaS => aaaS => aaabS => aaabbS => aaabbb

Explain this apparent contradiction in using Type3 grammar to generate a Type2 language.
(정규 문법에서 CFG 의 언어가 생성되는 명백한 모순에 대해 설명하라)

A.



정규 문법은 chomsky hierarchy 에 따르면 type3 grammar 로 type0,1,2,3 모든 grammar 의 범주에 속해 있다. 그 중 type2 문법인 CFG(Context-Free-Grammar)의 언어가 생성될 수 있는 것이다. 그리고 regular grammar 에서 생성되는 regular language 를 읽어 들이는 FA(Finite Automata)는 마찬가지로 PDA(Pushdown Automata)에 속해 있다. 따라서 regular grammar 에서 NPDA 가 읽어 들이는 Type2 의 언어가 생성될 수 있으며 이를 처리할 수 있게 되는 것이다.

4. Prove the following program for integer division:

```
{ x ≥ 0 ∧ y > 0}
  q := 0;
  r := x;
  while y ≤ r do
     begin
       r := r − y;
       q := q + 1;
     end
{y > r ∧ x = r + y * q}
```

A. 먼저 아래에서부터 loop invariant 인 I 를 찾는다

-> $\{y > r \land x = r + y * (q+1)\}$ q := q + 1 $\{y > r \land x = r + y * q\}$

-> $\{y > r - y \land x = r + y * q\}$ r := r – y$\{\{y > r \land x = r + y * (q+1)\}$

=> $\{y > r - 2y \land x = r + y * q\}$ q := q + 1, r := r – y $\{y > r - y \land x = r + y * q\}$

-> $\{y > r - 3y \land x = r + y * q\}$ q := q + 1, r := r – y $\{y > r - 2y \land x = r + y * q\}$

-> $\{y > r - 4y \land x = r + y * q\}$ q := q + 1, r := r – y $\{y > r - 3y \land x = r + y * q\}$

......

=> $\{y > r - ny \land x = r + y * q\}$ q := q + 1, r := r – y $\{y > r - (n-1)y \land x = r + y * q\}$

이때 loop 문으로 진입하면 $n > 0$ 이고 $r > 0$ 이므로 $n = r$ 로 치환하면
$\{I\} = \{y > r - ry \wedge x = r + y * q \wedge r > 0\}$이다.
그리고 loop 문을 통과 안 했을 때는 $n = 0$, 즉 $r \geq 0$ 이므로 $\{I\} = \{y > r - ry \wedge x = r + y * q \wedge r \geq 0\}$이다. 이 loop invariant 는 while 문의 앞에서도 성립, 안에서도 성립, 뒤에서도 성립하게 된다.
위의 loop 문을 검증하기 위해 post-condition 인 while 문의 조건 $y \leq r$ 을 $\{B\}$라 하여 $I \wedge B$, $I \wedge \neg B$ 가 true 가 되는지 확인한다.
한 단계씩 확인해 보면
$wp\{q := q + 1, y > r - ry \wedge x = r + y * q \wedge r \geq 0\}$
$= \{y > r - ry \wedge x = r + y * (q+1) \wedge r \geq 0\}$
$= \{y > r - ry \wedge x = r + y + y * q \wedge r \geq 0\}$
$wp\{r := r - y, y > r - ry \wedge x = r + y + y * q \wedge r \geq 0\}$
$= \{y > (r-1)(1-y) \wedge x = r - y + y + y * q \wedge r - y \geq 0)$
$= \{y > r-1-y(r-1) \wedge x = r + y * q \wedge r - y \geq 0\}$
$= \{y > r-1-y(r-1) \wedge x = r + y * q \wedge r \geq y\}$
위의 두 결과에 따라서 $I \wedge B$ 가 성립하는 것을 알 수 있다.
다음으로 $I \wedge \neg B$ 를 확인해 보면
$\{(y > r \wedge x = r + y * q) \wedge r < y\}$이므로 앞의 $y > r$ 과 뒤의 $r < y$ 가 동일 하므로 당연히 성립한다.
이를 통해서 위의 loop 문은 성립한다는 것을 알 수 있다.

다음으로 위의 값을 대입해서 확인해 보면
$wp(q := 0, y > r - ry \wedge x = r + y + y * q \wedge r \geq 0)$
$= \{y > r - ry \wedge x = r + y \wedge r \geq 0\}$
$wp(r := x, y > r - ry \wedge x = r + y \wedge r \geq 0)$
$= \{y > x - xy \wedge x = x + y \wedge x \geq 0)$
$= \{y > x(1-y) \wedge y \wedge x \geq 0\}$
-> y 는 $x \geq 0$ 이면 항상 $x(1-y)$보다 크며 가장 클 때는 $x = 0$ 일 때이므로 $y > 0$ 이 되는 것이
가장 strong 하다.
따라서 $\{y > 0 \wedge x \geq 0\}$을 유도할 수 있다.
위의 과정을 통해 나눗셈 프로그램이 정상 작동한다는 것을 확인할 수 있었다.