Programming Languages: Recitation 1

Goktug Saatcioglu

01.31.2019

1 Grammar

- Do this part fast (examples are more important).
- A grammar G is a tuple (Σ, N, P, S) where
  - Σ is the set of terminal symbols, a.k.a the alphabet,
  - N is the set of non-terminal symbols,
  - S is the root or start symbol,
  - P is the set of re-write rules (or productions) (or a transition function) in the form

    $$ABC \cdots \to XYZ \ldots$$

    where A, B, C, X, Y, Z are terminal and non-terminal symbols.
  - We require that $\Sigma \cap N = \emptyset$ (i.e. they are disjoint).
- A sequence of terminal and non-terminals is called a string.
- The language of a grammar is the set of all strings generated using the re-write rules of the grammar.

2 Regular expressions

- Describe a regular language over an alphabet Σ.
- We say that the regular expression R denotes the regular language $\mathbb{R}$.
- We have the following rules:
  - ϵ denotes $\emptyset$ (i.e. the empty language),
  - a denotes $\{a\}$ where $a \in \Sigma$ (i.e. the language that contains the string a),
  - RS denotes $\{ab \mid a \in \mathbb{R},\, b \in \mathbb{S}\}$ (i.e the concatenation of a string in R and a string in S),
  - $R \mid S$ denotes $\mathbb{R} \cup \mathbb{S}$ (i.e. the union of the sets of strings described by R and S, a.k.a alternation),
  - R^* denotes the concatenation of zero or more strings from $\mathbb{R}$ (known as Kleene star),
  - R^+ denotes the concatenation of one or more string from $\mathbb{R}$ (known as Kleene plus) [question: what is another way to express R^+, answer: $R^+ = RR^*$],
  - $R^?$ denotes $\epsilon \mid R$,
  - use paranthesis.
- Practice: Given the alphabet $\{a, b, c\}$ give a regular expression for the following (don't do all of them, 2 max):

– The language that contains the string aaa at some point. Answer: $(a \mid b \mid c)^* aaa (a \mid b \mid c)^*$.
– The language that must end with either the string a or the string bc. Answer: $(a \mid b \mid c)^*(a \mid bc)$.
– The language that does not contain the string cab. Answer: $(a \mid b \mid cb \mid ca(a \mid c))^*(\epsilon \mid c \mid ca)$. (Hard, ignore this).

3 Context-free grammars

- Do the example for arithmetic expressions.
- Ambigious one given below (do 2 different parse tree derivations).

$$x \in \mathbb{Z}$$

$$E ::= EOE \mid (E) \mid x$$

$$O ::= + \mid - \mid \times \mid \div$$

Derive the string "$3 \times 2 + 4$" in two different ways. Why is this a problem?

- The problem of detecting ambiguity is undecidable. Thus, we must reason by looking at the grammar to find an unambigious version. Any ideas? Notice that $\times$ and $\div$ have higher precedence thus this serves as our hint. We want $\times$ and $\div$ to bind stronger.
- Almost unambigious version given below.

$$x \in \mathbb{Z}$$

$$E ::= EAE \mid T$$

$$A ::= + \mid -$$

$$T ::= TMT \mid F$$

$$A ::= \times \mid \div$$

$$F ::= x \mid (E)$$

- Again the grammar is ambigious for the string "$3 + 2 + 4$". This is also a problem, why? Arithmetic operations are not usually associative because of potential arithmetic overflows (bounded representation of integers, floats). Thus, we want to parse the operations in a specific order: left-associative vs. right associative ($1+2+3 = (1+2)+3$ vs. $1+2+3 = 1+(2+3)$). We make the grammar left-associative in this case. [Achieve this by replacing the right side of each binary expression by the base case of that expression type.]

$$x \in \mathbb{Z}$$

$$E ::= EAT \mid T$$

$$A ::= + \mid -$$

$$T ::= TMF \mid F$$

$$A ::= \times \mid \div$$

$$F ::= x \mid (E)$$

- Finally, we can use a regular expression to replace the statement $x \in \mathbb{Z}$. What would this look like? Answer: $(\epsilon \mid -)(0 \mid (1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)(0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)^*)$. Can also be achieved using a context free grammar too (see lecture notes).