CS 241, Lecture 11 - Top Down Parsing, First and Follow

Thurs, Feb 14, 2019

1 Top Down Parsing

- Given a CFG $G = (N, \Sigma, P, S)$ and a terminal string $w \in \Sigma^*$, we want to find the derivation - the steps s.t. $S \Rightarrow \cdots \Rightarrow w$ or prove that $w \notin L(G)$ (just throw an error if so).
- Top-down parsing says start from S and try to get to w.
- Bottom-up parsing says says to start with w and try to see how we could get to w in the first place.
- We will first consider top-down parsing:
  - We start with S and store itermediate derivations in a stack, then match characters to w.
  - Use a stack to do so.
  - Every time we pop from stack, the consumed input and the reverse of the stack is equal to an intermediate step in our derivation.
  - We augment our grammar top include $\vdash$ and $\dashv$ to symbolize the beginning and end of the file, respectively. We also use S' to indicate a new start state.
  - We can see the algorithm described as follows:

Algorithm 1 Top-Down Parsing

```
Push S' onto the stack
while stack is non-empty do
    α = pop from stack
    if α ∈ N ∪ {S'} then
        Push symbols of β of a valid production rule α → β in reverse order on the
        stack (note derivation)
    else
        c = read_char()
        if c ≠ α then
            Reject
        end if
    end if
end while
if read_char() = EOF then
    Accept
else
    Reject
end if
```

– For example:

$$S \rightarrow AcB(1)$$
$$A \rightarrow ab(2)$$
$$A \rightarrow ff(3)$$
$$B \rightarrow def(4)$$
$$B \rightarrow ef(5)$$

Determine if $w = abcdef \in L(G)$. We add a rule:

$$S' \rightarrow \vdash S \dashv (0)$$

We want to look for $w = \vdash abcdef \dashv$ in this augmented grammar. This gives us the following parse table:

Stack	Read	Processing	Action
S'	ϵ	⊢ abcdef ⊣	Pop S', push ⊣, ⊢ (Rule 0)
⊣ S ⊢	ϵ	⊢ abcdef ⊣	Match ⊢
⊣ S	⊢	abcdef ⊣	Pop S, push B, c, A (Rule 1)
⊣ BcA	⊢	abcdef ⊣	Pop A, push b, a (Rule 2)
⊣ Bcba	⊢	abcdef ⊣	Match a
⊣ Bcb	⊢ a	bcdef ⊣	Match b
⊣ Bc	⊢ ab	cdef ⊣	Match c
⊣ B	⊢ abc	def ⊣	Pop B, push f, e, d (Rule 4)
⊣ fed	⊢ abc	def ⊣	Match d
⊣ fe	⊢ abcd	ef ⊣	Match e
⊣ f	⊢ abcde	f ⊣	Match f
⊣	⊢ abcdef	⊣	Match ⊣
ϵ	⊢ abcdef ⊣	ϵ	Accept, as stack = input = ϵ

– When we popped A, we had multiple possible choices - which rule would we use?

– We construct a predictor table, using a single character that we look at, to tell us which rule to use. For our example, for A, if we see an a, then we use rule 2; if we see a f then use rule 3.

– But this won't work if we have an element that contains more than one other element - for example, if we add a new rule $A \rightarrow adf$.

2 First and Follow

- A $LL(1)$ grammar is one that each cell of the predictor tables contains at most **one** entry.
- With an $LL(1)$ grammar, we can drop the set notation from the predictor table.
- We call it $LL(1)$ as:
  - First L: Scan left to right
  - Second L: Leftmost derivations
  - Number of symbols in lookahead: 1
- Constructing the lookahead table - we define four functions:
  - *Nullable*(β) =true iff $\beta \Rightarrow^* \epsilon$ and false otherwise
  - *Follow*$(A) = \{b \in \Sigma' : S' \Rightarrow^* \alpha A b \beta$ for some $\alpha, \beta \in V^*\}$
  - *Predict*$(A, a) = \{A \to \beta : a \in$ *First*$(\beta)\}$
  - *First*$(\beta) = \{a \in \Sigma` : \beta \Rightarrow^* a\gamma$, for some $\gamma \in V^*\}$
- More informally:
  - *Nullable*(β): boolean function, for $\beta \in V^*$ is true iff $\beta \Rightarrow^* \epsilon$.
  - *Follow(A)*: for any $A \in N'$, this is the set of elements of Σ' that can come immediately after A in a derivation starting from S'.
  - *Predict(A, a)*: production rules that apply when $A \in N'$ is on the stack, and $a \in \Sigma'$ is the next input character.
  - *First*(β): set of characters that can be the first letter of a derivation starting from $\beta \in V^*$.
- Note our definition of *Predict* is not correct right now.
- Our predict table looks like this:

	$\vdash$	a	b	c	d	e	f	$\dashv$
S'	{0}							
S		{1}					{1}	
A		{2}					{3}	
B					{4}	{5}		{**6**}

- We can see *First* with our previous example:

$$\begin{array}{llll} S' & \rightarrow & \vdash S \dashv & (0) \\ S & \rightarrow & AcB & (1) \\ A & \rightarrow & ab & (2) \\ A & \rightarrow & ff & (3) \\ B & \rightarrow & def & (4) \\ B & \rightarrow & ef & (5) \\ B & \rightarrow & \epsilon & (6) \end{array}$$

- Follow(S') = {} (Always!)
- Follow(S) = $\{\dashv\}$
- Follow(A) = $\{c\}$
- Follow(B) = $\{\dashv\}$

- We say that a $\beta \in V^*$ is **nullable** iff *Nullable*(β) = true
- Redefine *Predict*$(A, a) = \{A \rightarrow \beta : a \in$ *First*$(\beta)\} \cup \{A \rightarrow \beta : \beta$ is nullable and $a \in$ *Follow(A)*$\}$
- Note this ALL only works well for $LL(1)$. We will make our grammar work with $LL(1)$, and thus, these rules!