course: CSC 135-01 - Computing Theory and Programming Languages

instructor: Ted Krovetz

related_notes: 2022-04-05

Parsing

W14.2 | Tuesday, April 5, 2022 | 09:03 AM

Two Possible Goals For Parsing

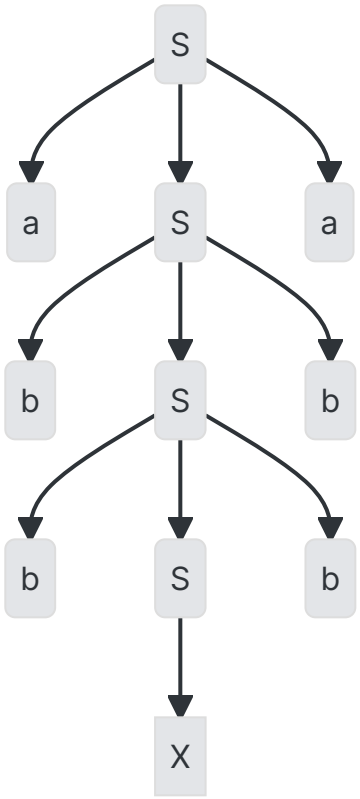
1. **Build a parse tree**; or
2. **Recognize** that a parse tree exist
    1. Simple goal is to ask "is this parse-able"

Recognizing

Grammar: $S \rightarrow aSa \,|bSb|x$
Language: abbxbba

Denervation from the string `abbxbba`:

- **Terminals** are lowercase letters
- **Nonterminal** are uppercase letters



Algorithm For Pushdown Automata (PDA) Parsing

```
while Stack not empty:
        top = pop
        token = next # Reads what's next in token string, but does not remove
from token string
        if top is a terminal:
                match(top) # Consumes next input if they match, otherwise
error
        elif top is nonterminal:
                push production right hand side

if input is empty:
        print("accept")
else:
        print("reject")
```

Further abstraction

```
while Stack not empty:
        top = pop
        token = next # Reads what's next in token string, but does not remove
from token string
        if top is a terminal:
```

```
                match(top) # Consumes next input if they match, otherwise
error
        elif top == A_subOne and token ∈ first(w_subOne) # A_1 --> w_1
                push(w_subOne)
        elif if top == a_subTwo and token ∈ first(w_subTwo) # a_2 --> w_2
                pust(w_subTwo)
        ⋮
        ⋮
        ⋮

if input is empty:
        print("accept")
else:
        print("reject")
```

For each `elif` $A_i \rightarrow w_2$

Only works if (cant read his hand writing) has only one production for the current state

Parsing PDA Code

Grammar: $S \rightarrow aSa\,|bSb|x$

```
While stack not empty:
    top = pop
        token = next
        if top == a,b,x:
                match(top)
        elif top == S and token == a:
                push(aSa)
        elif top == S and token == b:
                push(bSb)
        elif top == S and token == x:
                push(x)
        else:
                error
if input is empty:
        print("accept")
```

Which production to chose?

First(A) is the set of all first terminals of all strings derived form A

$A \rightarrow aA \rightarrow aB \rightarrow \ldots \rightarrow abb$

First(A) = $\{c \mid cs \in L(A) \; where \; 'c' \; is \; a \; terminal \; and \; 's' \; is \; a \; string \; of \; terminals\}$

$S \rightarrow aSa \mid bSb \mid x$

- $S \rightarrow aSa \rightarrow \ldots \; all \; start \; with \; 'a'$
- $S \rightarrow aSa \rightarrow \ldots \; all \; start \; with \; 'b'$
- $S \rightarrow x \rightarrow \ldots \; all \; start \; with \; 'x'$
- First(aSa) = {a}
- First(bSb) = {b}
- First(x) = {x}

Example 01

$A \rightarrow aA \mid \lambda$
$B \rightarrow aB \mid \lambda$
$S \rightarrow AB$

$First(a) = a$

- $A \rightarrow aA \ldots \quad a \in$ First(A) doesn't have first terminal
- $First(A) \rightarrow \lambda$

$First(B) = b$
$First(S) = a, b$

- $s \rightarrow AB \rightarrow aAB \rightarrow \ldots a \in First(S)$
- $s \rightarrow AB \rightarrow B \rightarrow \ldots b \in First(S)$
- $S \rightarrow AB \rightarrow B \rightarrow \lambda$

To Find Sets

1. Identify all set constraints
2. Seed the first sets with the $\in$ constraints
3. Satisfy **C** constraints by copying anything in the Left Hand Side missing in the Right Hand Side
4. Repeat step 3 until nothing happens

Methodical way to find First

If you have...	you can deduce
$A \rightarrow \lambda$	nothing

If you have...	you can deduce
$A \rightarrow xw$	$x \in \ First(A)$
$A \rightarrow Bw$	$First\,(B) \leq First\,(B)$
$A \rightarrow Bw$ and $\lambda \in L\,(B)$	$First\,(w) \leq First\,(A)$