1 LL(1) Parsing

1.1 Introduction

- **LL(1) parsing:** Top-down parsing algorithm which creates a left-to-right leftmost derivation requiring only 1 symbol of lookahead
  - Requires a grammar which is:
    - Unambiguous
    - Left-factored
    - Free of left-recursion
  - Only one possible production can exist given the next token
- Grammar validity:
  - A grammar is LL(1) if and only if, for each pairing of a non-terminal symbol with a terminal symbol, only one possible production rule can be applied
  - A grammar is not LL(1) if and only if there exists at least one pairing of a non-terminal symbol and a terminal symbol which can expand to multiple production rules (e.g. $A \to bc; A \to bd$)

1.2 Predictive Parsing Tables

- **Predictive parsing table:** Table which correlates the leftmost non-terminal symbol and the next terminal symbol with the only possible production
  - No cell can have more than one string as the expansion to the rule
  - Empty cells mean parsing errors
  - Uses lookahead to resolve conflicts between multiple possible production rule applications for the same symbol
  - Layout: See figure 1

Figure 1: Predictive Parsing Table Layout

		Terminal symbols	$
Non-terminal Symbols	*Symbol*	*Production result*	

- To create a predictive parsing table:
  - For each non-terminal symbol X on the left column:
    - If there is a production rule where X expands to epsilon (ϵ), then:
      - For each terminal symbol which is in the $FOLLOW$ of symbol X, write ϵ
    - If there is a production rule where X expands to a string which is not epsilon (ϵ), then:
      - For each terminal symbol which is in the $FIRST$ of symbol X, write the expansion of the valid production rule
- E.g. In figure 2, $Y \to \epsilon$ is the valid production rule to be chosen when $+,)$, or $\$$ are the lookahead symbol

Figure 2: LL(1) Conflict Resolution with $FOLLOW$

Production Rules	$E \to TX$
	$X \to \epsilon$
	$X \to +E$
	$T \to (E)$
	$T \to idY$
	$Y \to *T$
	$Y \to \epsilon$
$FOLLOW(Y)$	$= FOLLOW(T)$
	$= (FIRST(X) - \{\epsilon\}) + FOLLOW(E)$
	$= \{+,), \$\}$

1.3 LL(1) Parsing Trace

- LL(1) parsing trace:
  - Stack is used to keep track of non-terminal symbols
  - Stack and input strings are terminated with the end of input symbol ($)
  - Layout: See figure 3

Figure 3: LL(1) Parsing Trace Example

Stack	**Input**	**Action**
string $	*string $*	*Production rule to expand OR*
		ϵ *OR*
		Terminal α *OR*
		acc

- To create an LL(1) parsing trace:
  - For the leftmost input symbol and the leftmost stack symbol:
    - If the leftmost stack symbol is a non-terminal symbol, then use the leftmost input and stack symbols to find the corresponding production rule expansion from the predictive parsing table, and:
      - If the expansion is not an epsilon, then replace the leftmost stack symbol with the expansion
      - If the expansion is an epsilon (ϵ), then remove the leftmost stack symbol
    - If the leftmost stack symbol is a terminal symbol and the leftmost stack and input symbols match, then consume both of them with the action *Terminal*
    - If the leftmost stack and input symbols are the end of input ($), then accept the parse