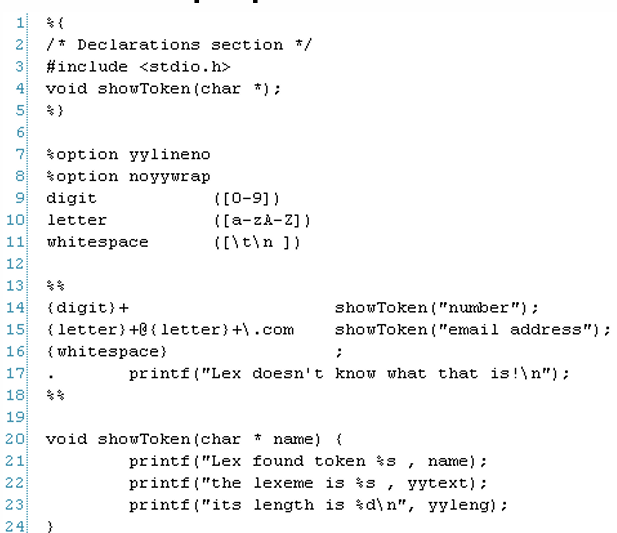
***Compiler:***

***Lex -***

* *translates our code(lexeme’s) into ‘tokens’*
* *Finds lexical errors*
* *Filters redundant ‘spaces’*
* *Saves variable names and operation types for the* ***Semantic Analizer***

*Lex File Structure:*

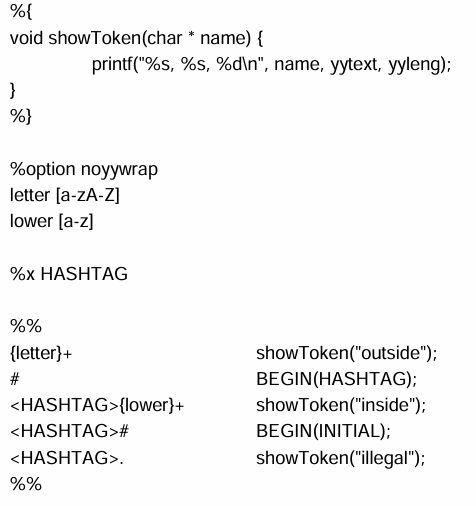


*yylex returns only when it gets to a ‘return’ token OR it got to the end of the input*

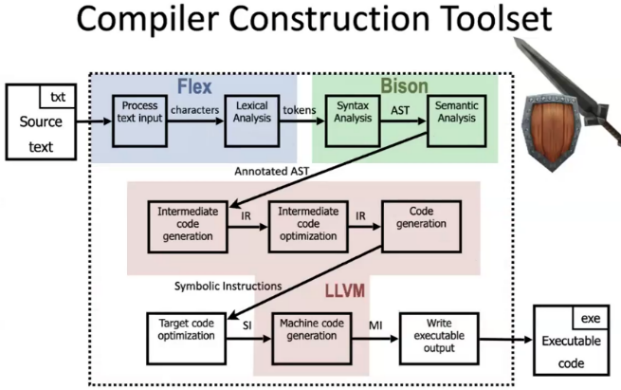
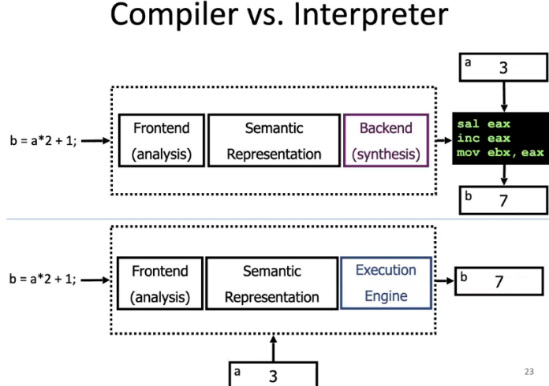
*lex error occurs only when the lexer can’t identify the token of the lexeme*

*in case of a Conflict(can’t decide which token to choose):*

* *The lexer will always choose the longest fitting lexeme*
* *If the lengths are equal, the highest priority token(the one defined first) will be chosen*

*Start conditions - allow us to use selective rules for special “modes”:*

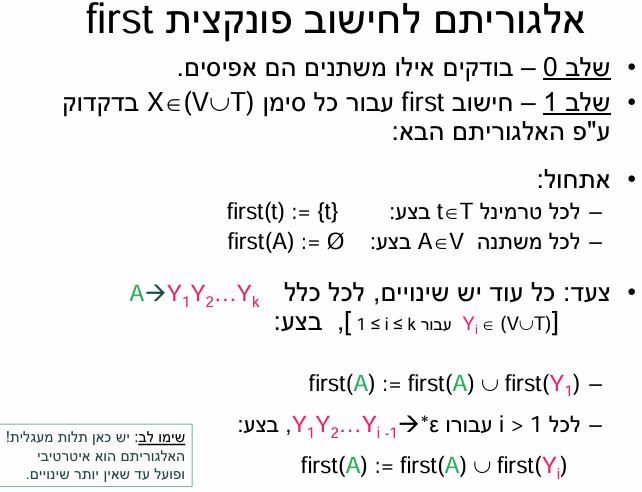




**Top-Down Parser** –

* begin with S
* apply derivation rules
* stop when the whole input is derived OR when you get stuck and can’t derive anymore

**LL(1)** **Parser** –

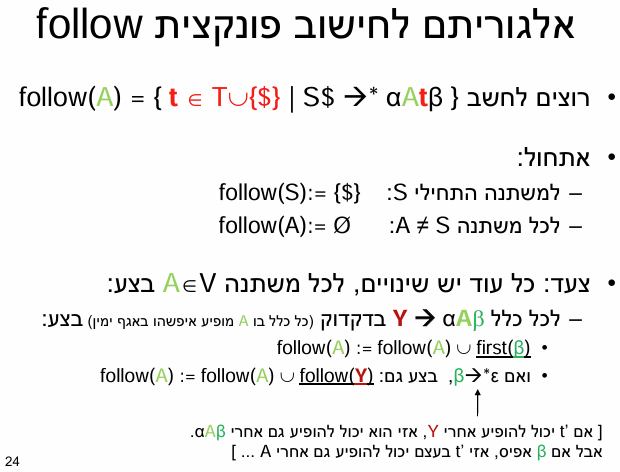
* uses **Select** to choose the next rule to derive
* reads the input from **Left** to right, creates a **Left** to right analysis with the help of **1** **terminal** lookahead

****

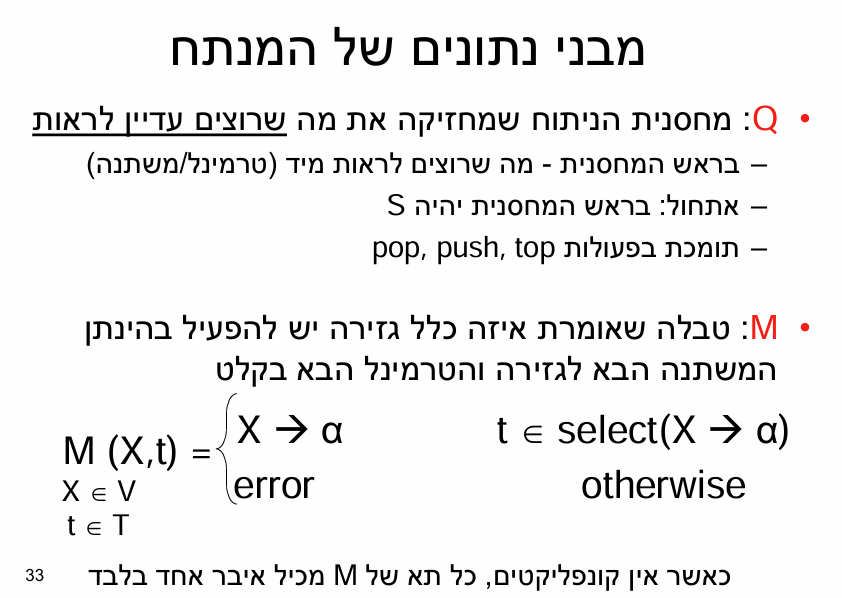
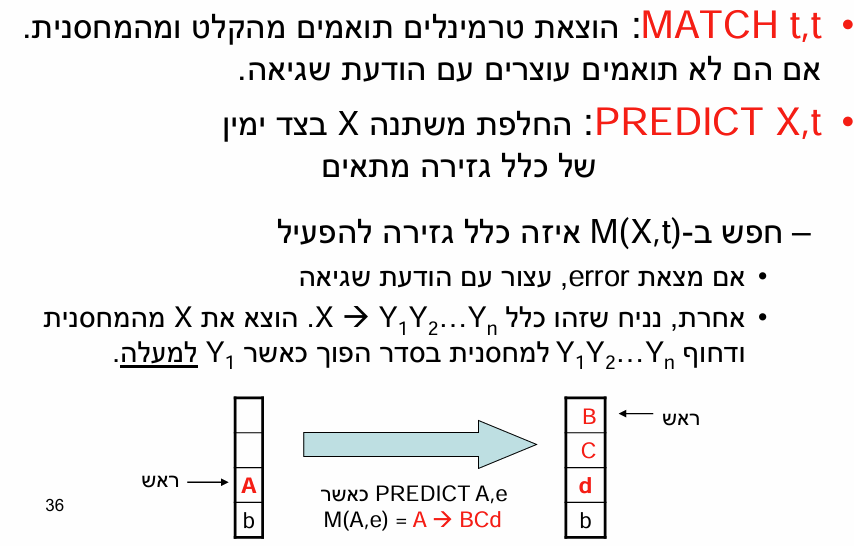
**A black text with black text

Description automatically generated with medium confidence**A white background with black text and numbers

Description automatically generated

****

Grammar G has analyzer for each two rules for the same variable :

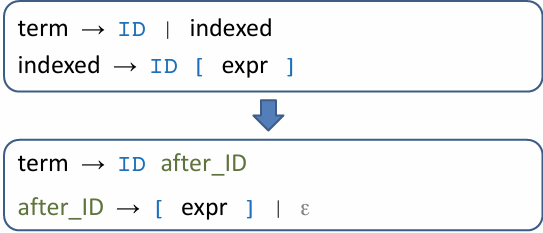
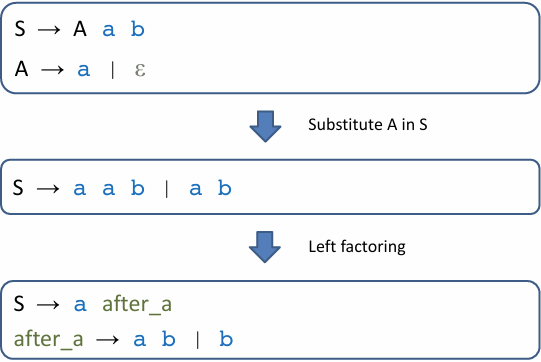
****

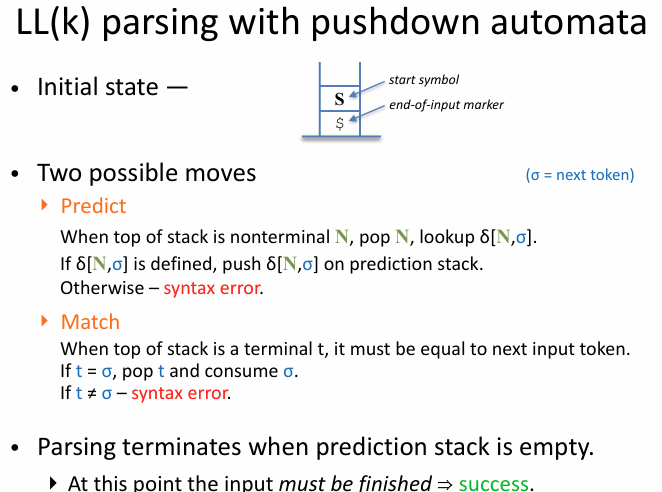
**A screenshot of a computer code

Description automatically generated**

*there exists some derivation sequence*

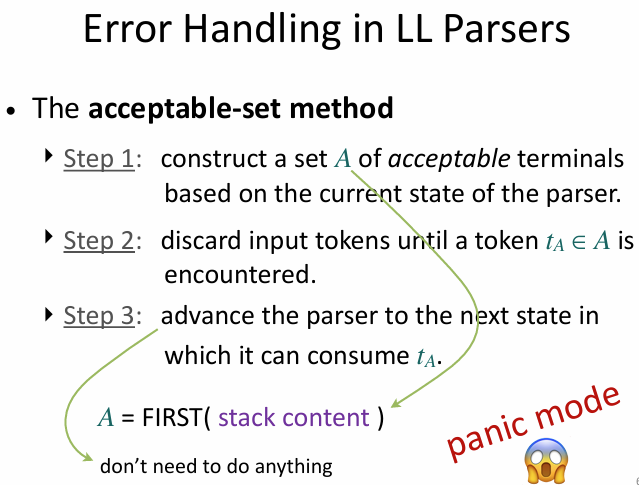
***LL(1) conflicts***

* ***First/First conflict –*** *Solved using Left Factoring*
* ***First/Follow conflict –*** *solved using Grammatical Substitution*



* ***A blue arrow pointing to the right

  Description automatically generatedLeft Recursion –*** *Left Recursion Removal*

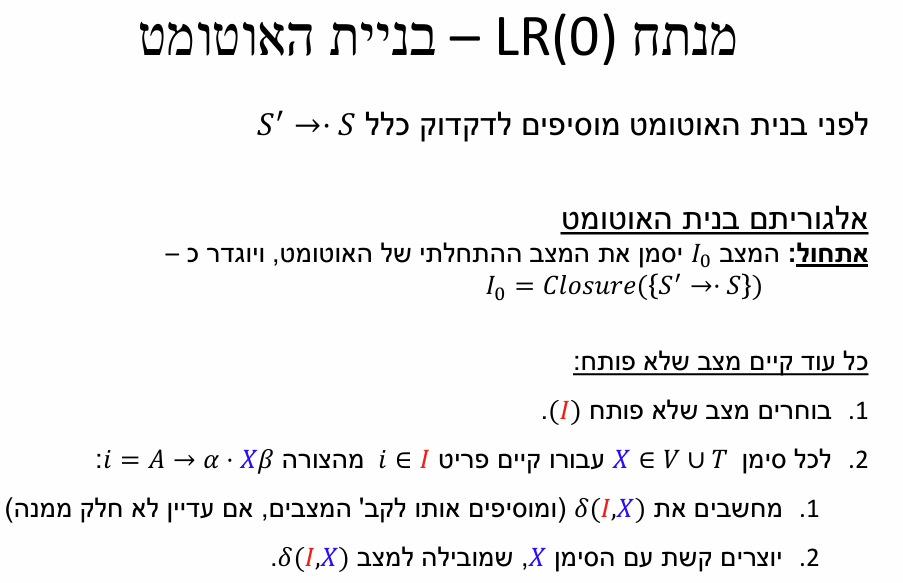
******

***A text on a white background

Description automatically generated***

***Bottom-up Parser***

* *Construct the rightmost derivation*
* *Apply rules from right to left*
* *Reduce the right-hand side of a production to its non-terminal*

***LR(K) Grammars –***

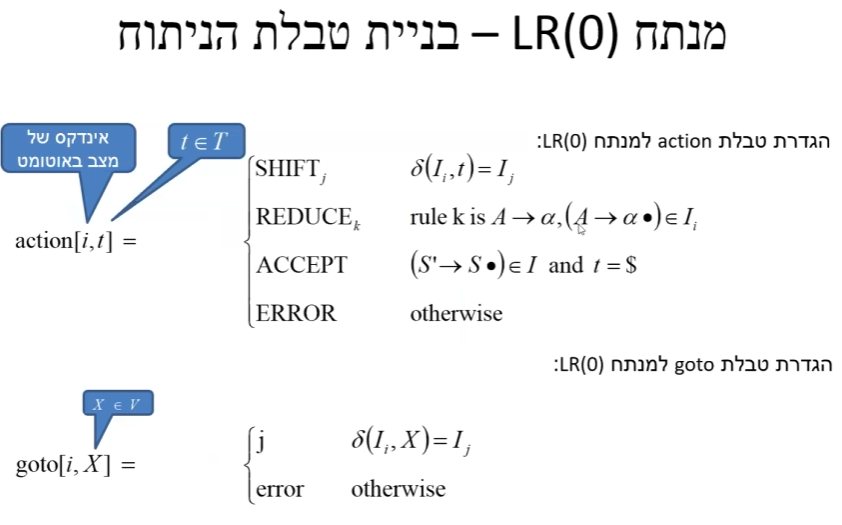
* *Bottom-Up analysis*
* *Scan input from left to right*
* *Producing the rightmost derivation*
* *Lookahead of K tokens*

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

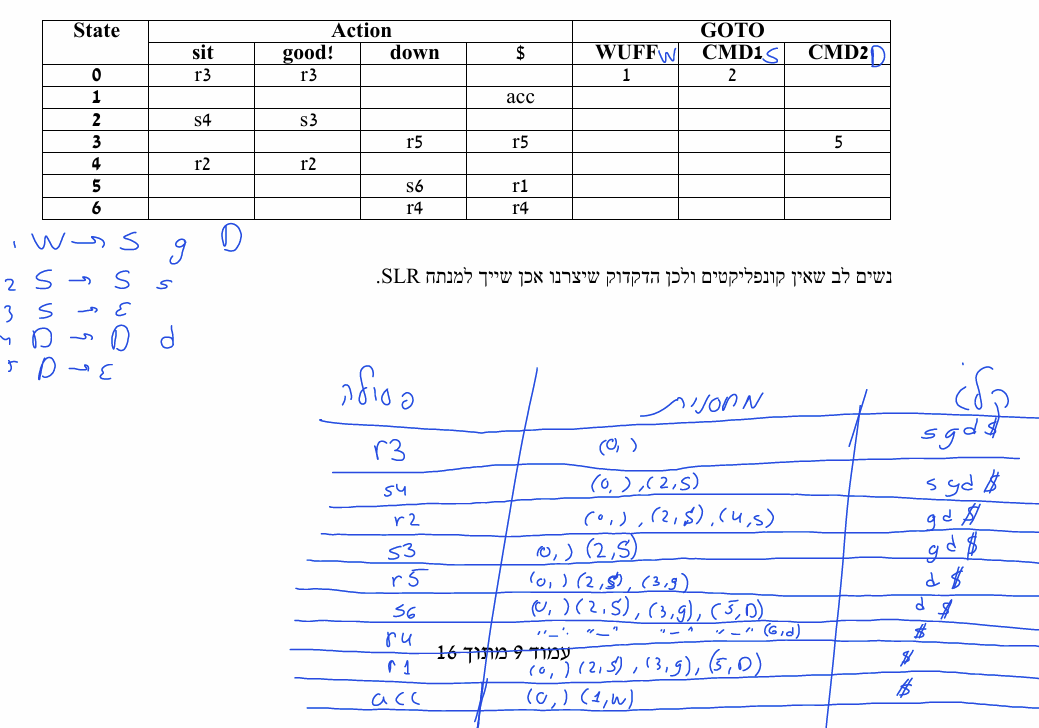
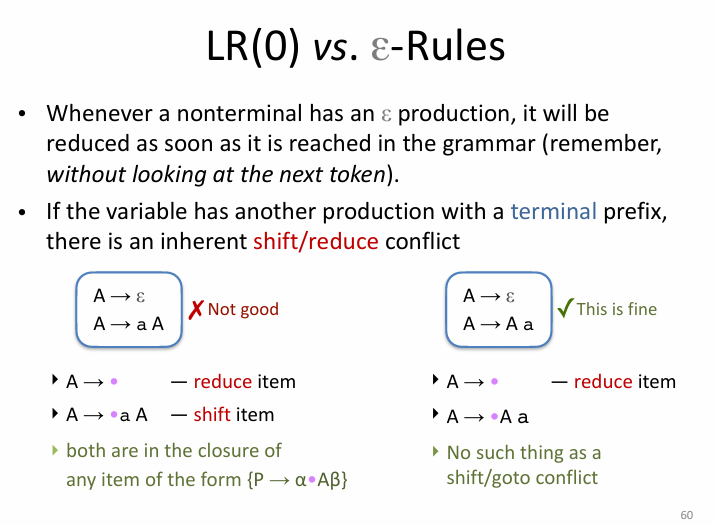
****



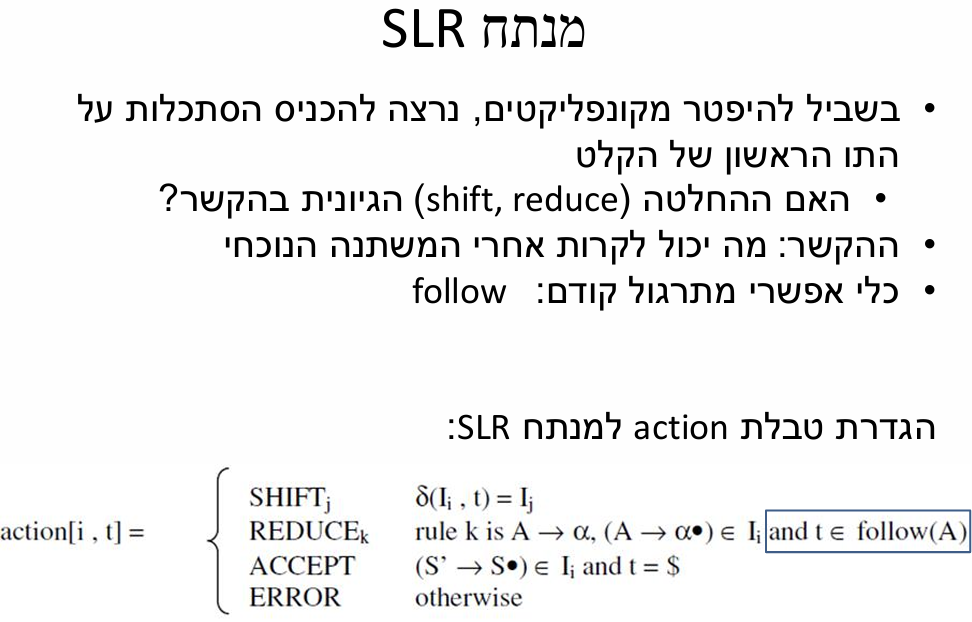
A black and white text

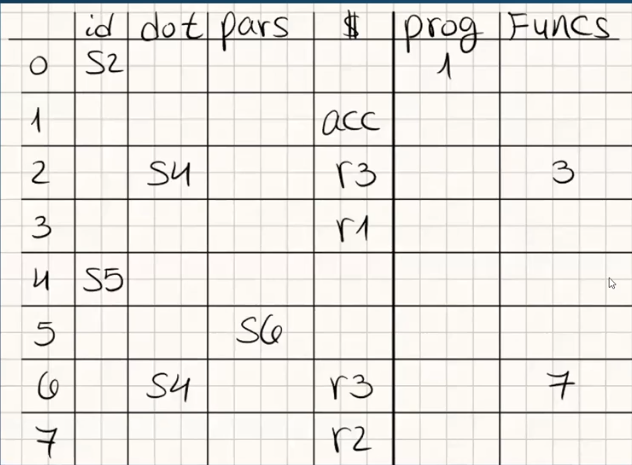
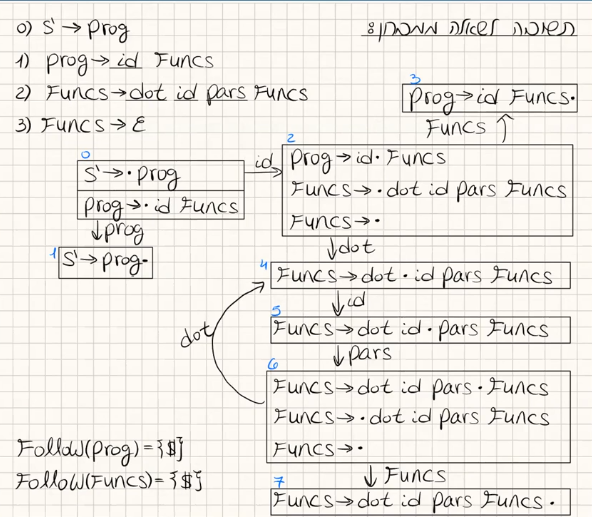
Description automatically generated

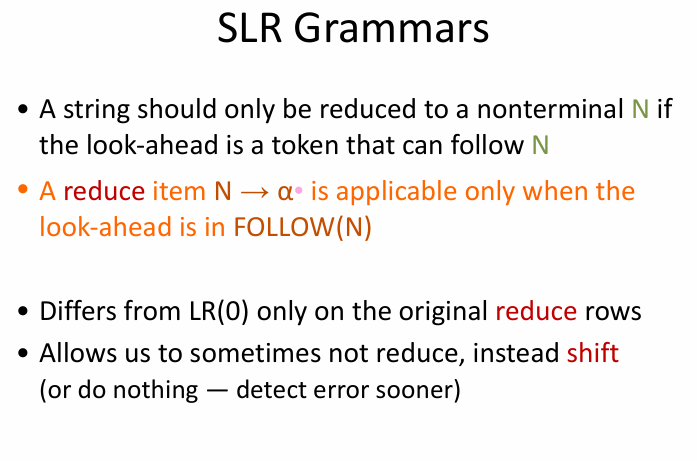
Every language in LR(0) is also in SLR

****

A white background with black text

Description automatically generated

****

****

A screenshot of a computer

Description automatically generatedA screenshot of a computer

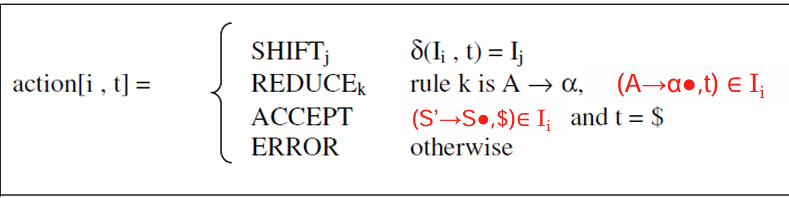
Description automatically generated

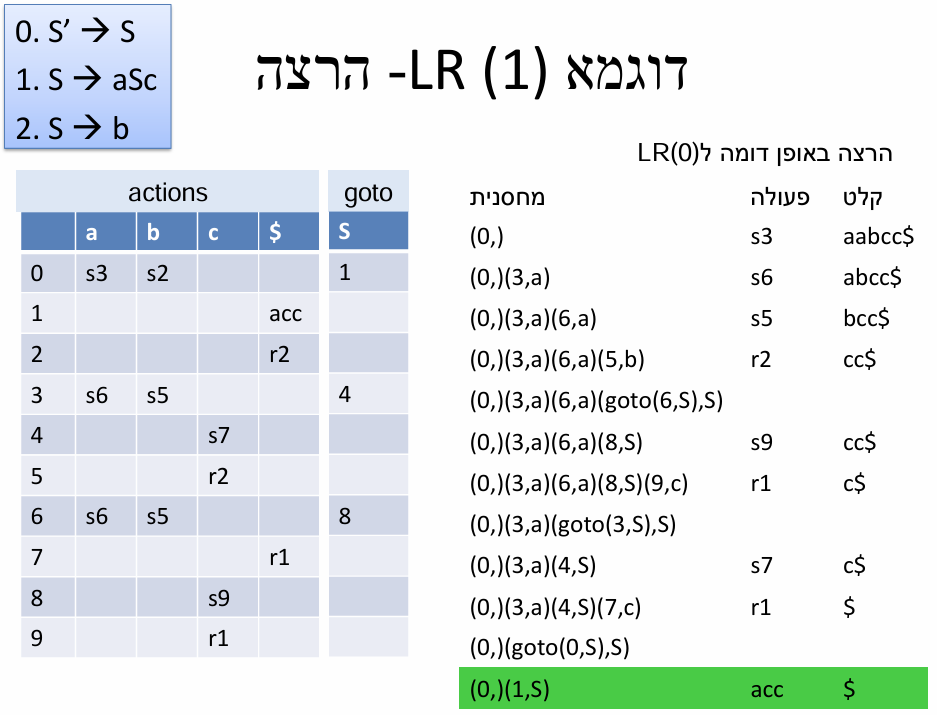
**A diagram of a flowchart

Description automatically generatedA screenshot of a computer

Description automatically generated**

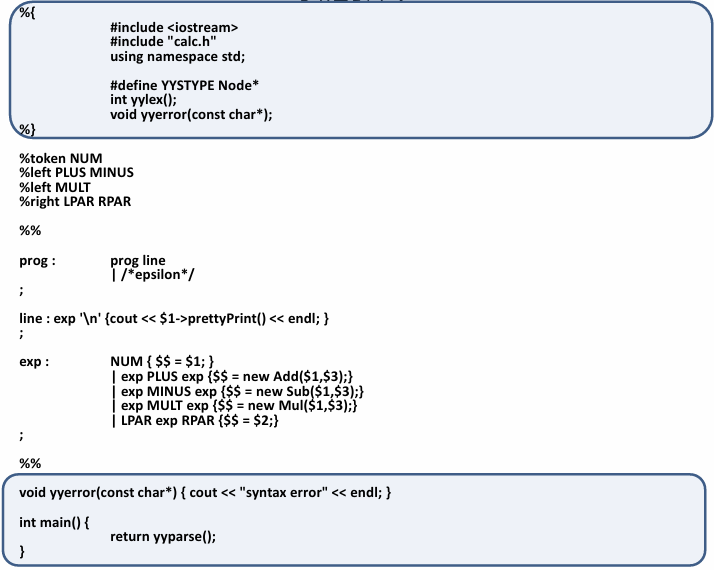
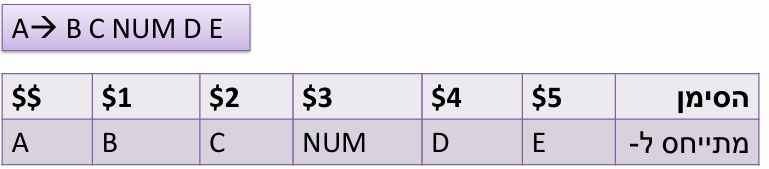
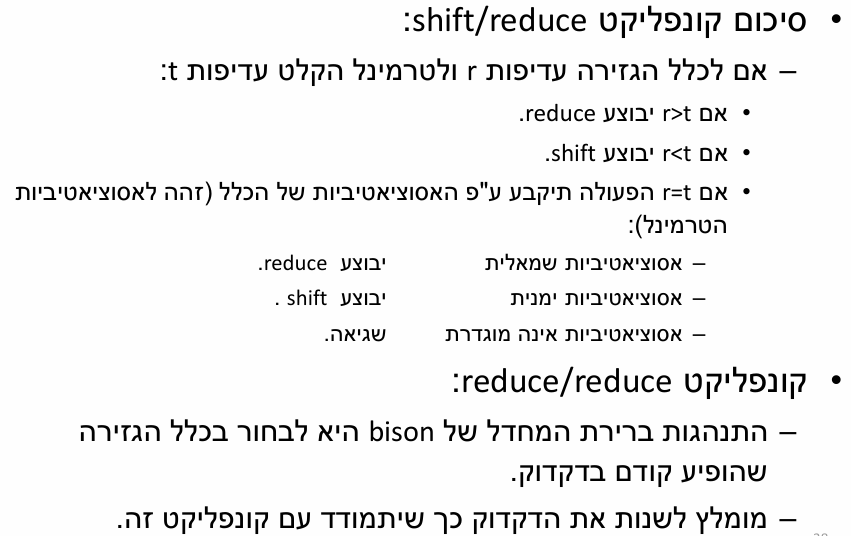


****

****

**A screenshot of a math problem

Description automatically generatedA screenshot of a computer

Description automatically generated**



**Semantic Analysis**

Allows us to do semantic checks during the syntax analysis

**Synthesized Attribute** – the attribute is dependent only on the value of the attributes of its sons

**Inherited Attribute** – the attribute is dependent only on the value of the Father & Brothers

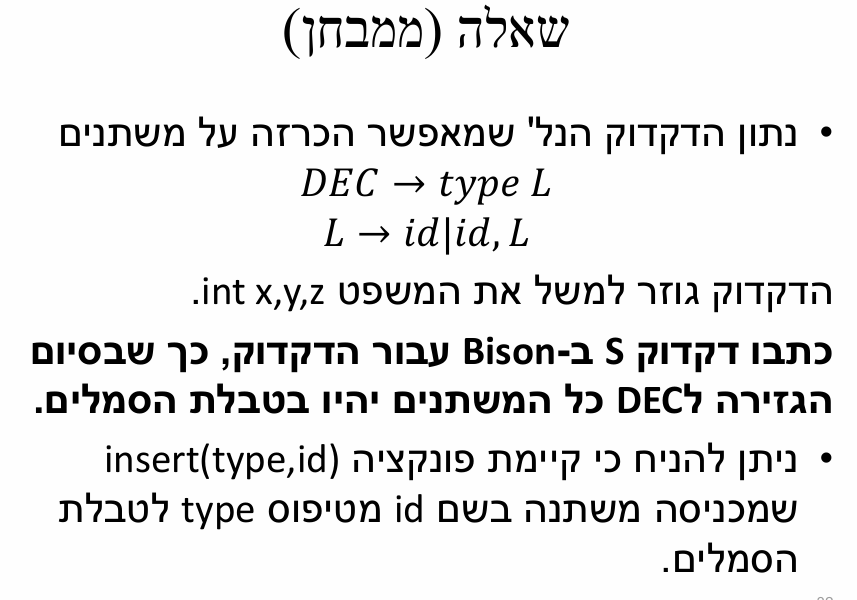
**S-attributed Grammars** – use only synthesized attributes

* **A screenshot of a computer

  Description automatically generated**A screenshot of a computer

  Description automatically generatedSemantic analysis is done during the construction of the AST in Bottom-Up parsers
* A white background with black text and colorful text

  Description automatically generated with medium confidenceA close up of a computer code

  Description automatically generatedNo need to parse the tree a 2nd time

**Symbol Table** – Data structure that holds symbols(var names, funcs etc), supports **static scoping** and supports efficient symbol lookup

**Data Flow Analysis**

**Liveness –**

* A variable **x** is considered **live** at a particular program point if its value will be **used later** before any new assignment is made to it.
* If a variable is assigned a new value before it is used, the previous value is **dead**.

**Peephole Matching -** local optimization technique that scans small sequences of instructions to identify and replace inefficient patterns with optimized code, Steps:

* **Expander** – Converts **IR → Lower-Level IR (LLIR)** with detailed operations.
* **Simplifier** – Uses forward substitution, algebraic simplification, local constant propagation, and dead-effect elimination
* **Matcher** – Translates **optimized LLIR → Assembly code** with minimal instruction cost.

**A blue line on a white background

Description automatically generatedLocal optimizations**

* A blue line on a white background

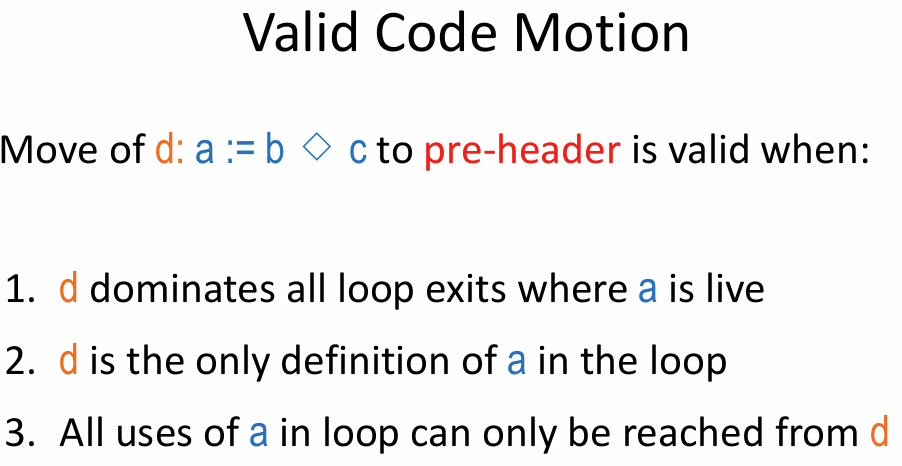
  Description automatically generated**Constant Propagation**
* A blue line with a white background

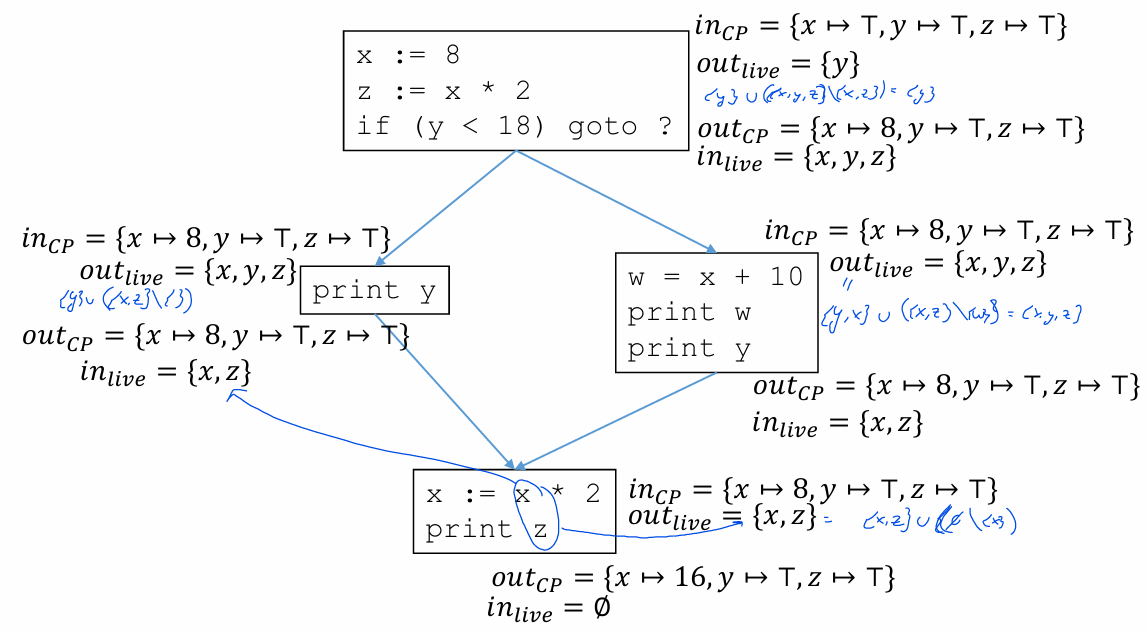
  Description automatically generated**Copy Propagation**
* **Common Subexpression Elimination**
* **Algebraic Simplification**
* **Constant Folding**
* **Useless code elimination**
  + **Dead Code:** Assigned but never used.
  + **Useless Assignments:** Rewriting a variable before its next use.
  + **Unreachable Code:** Code that is never executed due to jumps.

**Loop Optimization –**

1. identify loops (header, body, back edge)
2. Apply loop optimizations:
   1. **Loop invariant code** motion (move computations outside loops).
   2. **Strength reduction** of induction variable(convert expensive calculations into simpler ones).
   3. Induction **variable elimination** (remove unnecessary loop variables).

**Loop Detection**

* **Dominators** - node **u** **dominates** a node **v** in a **Control Flow Graph (CFG)** if **every path** from the start node to **v** must go through **u**.
  + **Header** – dominates loop body
  + **Back edge** – Target dominates source



A white card with black text and blue text

Description automatically generatedA close up of black text

Description automatically generated

A black and white rectangle with letters in it

Description automatically generatedA math equations and formulas

Description automatically generated

A close up of black text

Description automatically generated

**Global Optimizations:**

* A close up of a text

  AI-generated content may be incorrect.**Branch Chaining –** removal of useless ‘branch’ chains

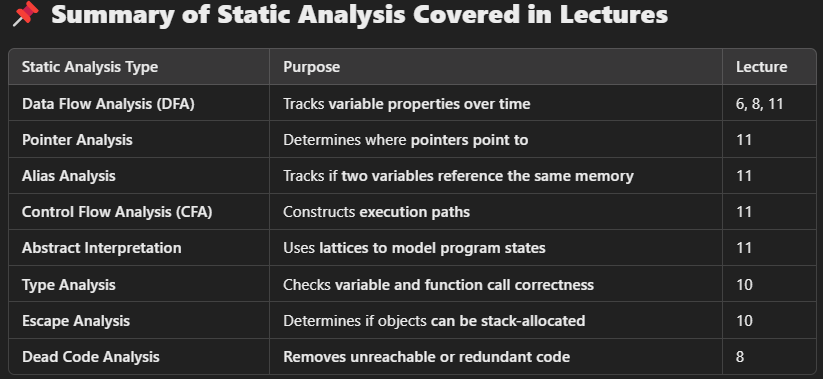
A blue arrow pointing to the right

AI-generated content may be incorrect.

* **A close up of a text

  AI-generated content may be incorrect.Dead Code -**  removal of an unreachable code block, non-optimal algorithm:
* A diagram of a block diagram

  AI-generated content may be incorrect.**Just In Time Compilation –** move some of the code that only runs once to be taken care of by the interpreter instead of the compiler



A screenshot of a computer

AI-generated content may be incorrect.

A black screen shot of a black table

AI-generated content may be incorrect.

**Lattices Explained Kinda:**

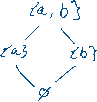
**–** greatest element of

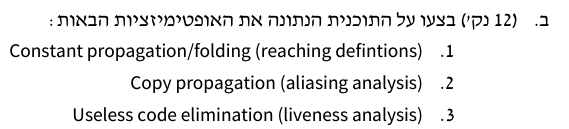
**–** least element of

**Lower Bound:**

**Upper Bound:**

Examples:





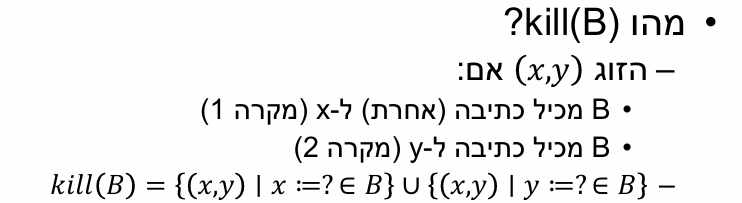


A group of black letters

AI-generated content may be incorrect.**A blue rectangular sign with black text

AI-generated content may be incorrect.Copy Propagation**:



**A blue rectangular sign with black text

AI-generated content may be incorrect.A blue rectangle with black text

AI-generated content may be incorrect.**

A close up of a sign

AI-generated content may be incorrect.A close up of black text

AI-generated content may be incorrect.**Liveness:**

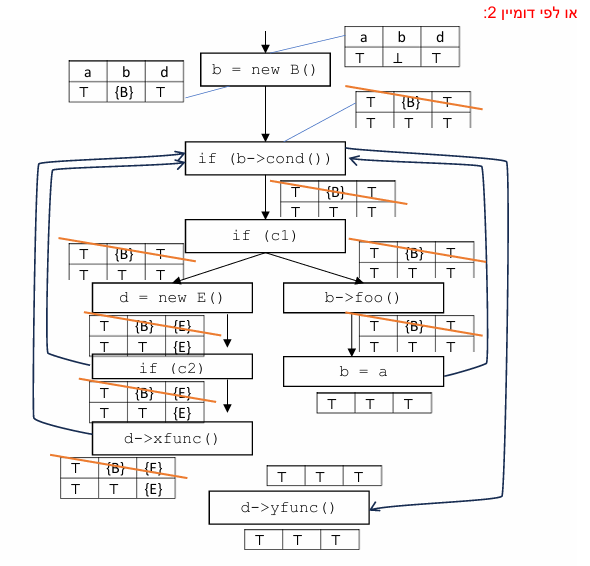


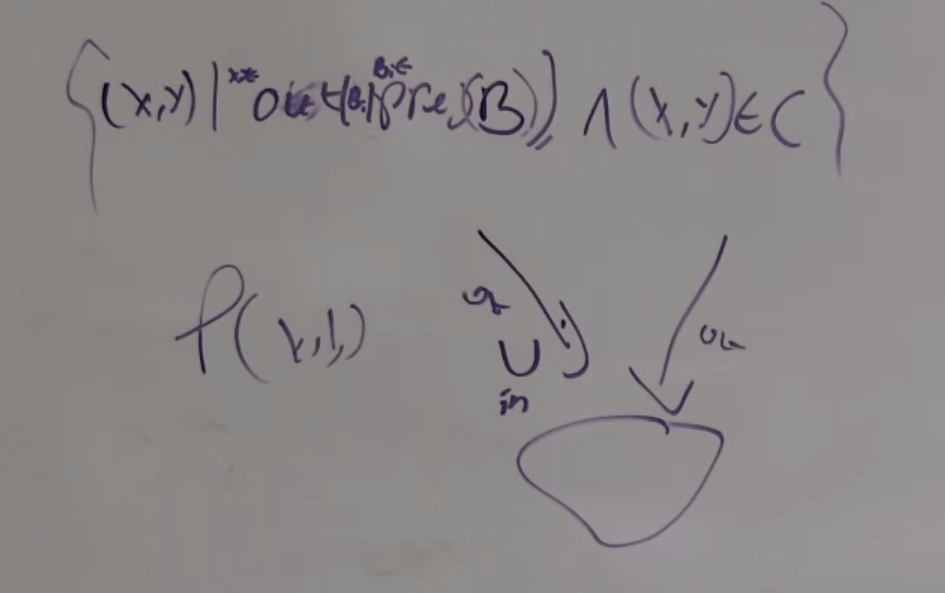
**Const Propagation/Folding(Reaching Definitions):**



A white paper with text and numbers

AI-generated content may be incorrect.





A white board with writing on it

AI-generated content may be incorrect.

