**Script** generated by TTT

Title: Petter: Compilerbau (02.05.2016)

Date: Mon May 02 14:25:31 CEST 2016

Duration: 90:26 min

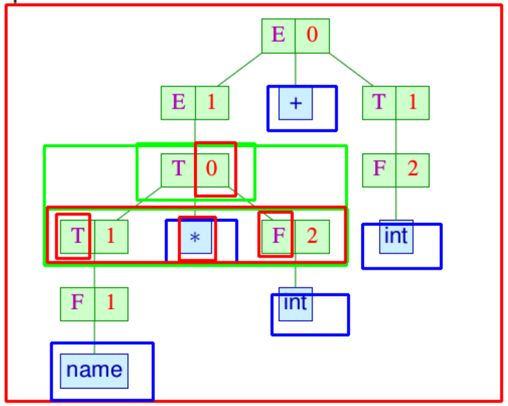
Pages: 41

## Derivation Tree

Derivations of a symbol are represented as derivation trees:

... for example:

$$
\begin{aligned}
\underline{E} &\rightarrow^0 \underline{E} + T \\
&\rightarrow^1 \underline{T} + T \\
&\rightarrow^0 T * \underline{F} + T \\
&\rightarrow^2 \underline{T} * \text{int} + T \\
&\rightarrow^1 \underline{F} * \text{int} + T \\
&\rightarrow^1 \text{name} * \text{int} + \underline{T} \\
&\rightarrow^1 \text{name} * \text{int} + \underline{F} \\
&\rightarrow^2 \text{name} * \text{int} + \text{int}
\end{aligned}
$$




A derivation tree for $A \in N$:

inner nodes: rule applications

root: rule application for $A$

leaves: terminals or $\epsilon$

The successors of $(B, i)$ correspond to right hand sides of the rule



## Special Derivations

**Attention:**

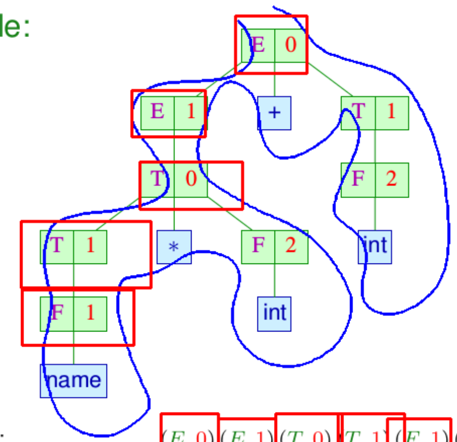
In contrast to arbitrary derivations, we find special ones, always rewriting the leftmost (or rather rightmost) occurance of a nonterminal.

- These are called leftmost (or rather rightmost) derivations and are denoted with the index $L$ (or $R$ respectively).
- Leftmost (or rightmost) derivations correspondt to a left-to-right (or right-to-left) preorder-DFS-traversal of the derivation tree.
- Reverse rightmost derivations correspond to a left-to-right postorder-DFS-traversal of the derivation tree



## Special Derivations

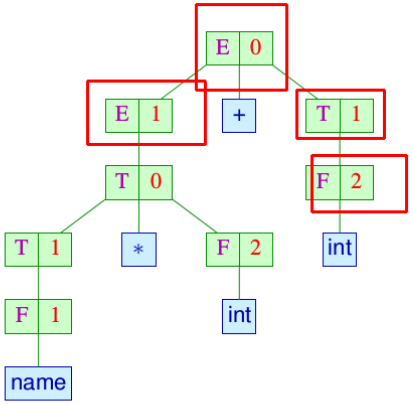
... for example:




Leftmost derivation: $(E,0)\,(E,1)\,(T,0)\,(T,1)\,(F,1)\,(F,2)\,(T,1)\,(F,2)$

## Special Derivations

... for example:



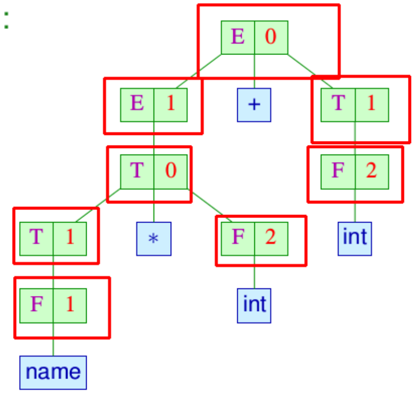

Leftmost derivation: $(E,0)\,(E,1)\,(T,0)\,(T,1)\,(F,1)\,(F,2)\,(T,1)\,(F,2)$

Rightmost derivation: $(E,0)\,(T,1)\,(F,2)\,(E,1)\,(T,0)\,(F,2)\,(T,1)\,(F,1)$



## Special Derivations

... for example:




Leftmost derivation: $(E,0)\,(E,1)\,(T,0)\,(T,1)\,(F,1)\,(F,2)\,(T,1)\,(F,2)$

Rightmost derivation: $(E,0)\,(T,1)\,(F,2)\,(E,1)\,(T,0)\,(F,2)\,(T,1)\,(F,1)$

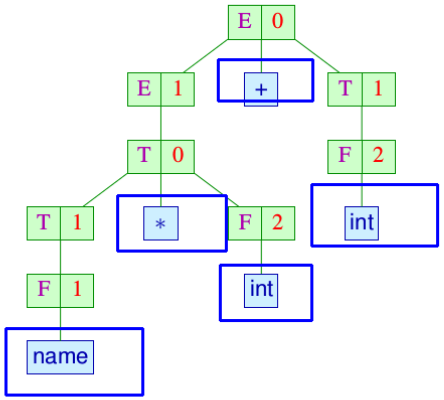
Reverse rightmost derivation: $(F,1)\,(T,1)\,(F,2)\,(T,0)\,(E,1)\,(F,2)\,(T,1)\,(E,0)$



## Unique Grammars

The concatenation of leaves of a derivation tree $t$ are often called yield($t$) .

... for example:




gives rise to the concatenation: name $*$ int $+$ int .



## Unique Grammars

**Definition:**

Grammar $G$ is called unique, if for every $w \in T^*$ there is maximally one derivation tree $t$ of $S$ with yield$(t) = w$.

... in our example:

| | | | | | | |
|---|---|---|---|---|---|---|
| $E$ | $\to$ | $E{+}E^{\,0}$ | $E{*}E^{\,1}$ | $(\,E\,)^{\,2}$ | name $^{3}$ | int $^{4}$ |
| $E$ | $\to$ | $E{+}T^{\,0}$ | $T^{\,1}$ | | | |
| $T$ | $\to$ | $T{*}F^{\,0}$ | $F^{\,1}$ | | | |
| $F$ | $\to$ | $(\,E\,)^{\,0}$ | name $^{1}$ | int $^{2}$ | | |

The first one is ambiguous, the second one is unique

## Conclusion:

- A derivation tree represents a possible hierarchical structure of a word.
- For programming languages, only those grammars with a unique structure are of interest.
- Derivation trees are one-to-one corresponding with leftmost derivations as well as (reverse) rightmost derivations.



## Conclusion:

- A derivation tree represents a possible hierarchical structure of a word.
- For programming languages, only those grammars with a unique structure are of interest.
- Derivation trees are one-to-one corresponding with leftmost derivations as well as (reverse) rightmost derivations.

- Leftmost derivations correspond to a top-down reconstruction of the syntax tree.
- Reverse rightmost derivations correspond to a bottom-up reconstruction of the syntax tree.





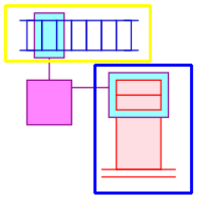
## Chapter 2:

## Basics of Pushdown Automata



## Basics of Pushdown Automata

Languages, specified by context free grammars are accepted by Pushdown Automata:



The pushdown is used e.g. to verify correct nesting of braces.

## Example:

**States:** $0, 1, 2$
**Start state:** $0$
**Final states:** $0, 2$

| | | |
|---|---|---|
| $0$ | $a$ | $11$ |
| $1$ | $a$ | $11$ |
| $11$ | $b$ | $2$ |
| $12$ | $b$ | $2$ |



## Example:

**States:** $0, 1, 2$
**Start state:** $0$
**Final states:** $0, 2$

| | | |
|---|---|---|
| $0$ | $a$ | $11$ |
| $1$ | $a$ | $11$ |
| $11$ | $b$ | $2$ |
| $12$ | $b$ | $2$ |

### Conventions:

- We do not differentiate between pushdown symbols and states
- The rightmost / upper pushdown symbol represents the state
- Every transition consumes / modifies the upper part of the pushdown



### Definition: Pushdown Automaton


Friedrich Bauer


Klaus Samelson

A pushdown automaton (PDA) is a tuple
$M = (Q, T, \delta, q_0, F)$ with:

- $Q$ a finite set of states;
- $T$ an input alphabet;
- $q_0 \in Q$ the start state;
- $F \subseteq Q$ the set of final states and
- $\delta \subseteq Q^+ \times (T \cup \{\epsilon\}) \times Q^*$ a finite set of transitions



### Definition: Pushdown Automaton


Friedrich Bauer


Klaus Samelson

A pushdown automaton (PDA) is a tuple
$M = (Q, T, \delta, q_0, F)$ with:

- $Q$ a finite set of states;
- $T$ an input alphabet;
- $q_0 \in Q$ the start state;
- $F \subseteq Q$ the set of final states and
- $\delta \subseteq Q^+ \times (T \cup \{\epsilon\}) \times Q^*$ a finite set of transitions

We define computations of pushdown automata with the help of transitions; a particular computation state (the current configuration) is a pair:

$$(\gamma, w) \in Q^* \times T^*$$

consisting of the pushdown content and the remaining input.

## ... for example:

**States:** $0, 1, 2$
**Start state:** $0$
**Final states:** $0, 2$

| | | |
|---|---|---|
| 0 | $a$ | 11 |
| 1 | $a$ | 11 |
| 11 | $b$ | 2 |
| 12 | $b$ | 2 |

$(0, \; a\,a\,a\,b\,b\,b) \vdash (11, \; a\,a\,b\,b\,b)$



## ... for example:

**States:** $0, 1, 2$
**Start state:** $0$
**Final states:** $0, 2$

| | | |
|---|---|---|
| 0 | $a$ | 11 |
| 1 | $a$ | 11 |
| 11 | $b$ | 2 |
| 12 | $b$ | 2 |

$(0, \; a\,a\,a\,b\,b\,b) \;\vdash\; (1\,1, \; a\,a\,b\,b\,b)$



## ... for example:

**States:** $0, 1, 2$
**Start state:** $0$
**Final states:** $0, 2$

| | | |
|---|---|---|
| 0 | $a$ | 11 |
| 1 | $a$ | 11 |
| 11 | $b$ | 2 |
| 12 | $b$ | 2 |

$$\begin{aligned}(0, \; a\,a\,a\,b\,b\,b) &\vdash (1\,1, \; a\,a\,b\,b\,b)\\ &\vdash (1\,1\,1, \; a\,b\,b\,b)\end{aligned}$$



## ... for example:

**States:** $0, 1, 2$
**Start state:** $0$
**Final states:** $0, 2$

| | | |
|---|---|---|
| 0 | $a$ | 11 |
| 1 | $a$ | 11 |
| 11 | $b$ | 2 |
| 12 | $b$ | 2 |

$$\begin{aligned}(0, \; a\,a\,a\,b\,b\,b) &\vdash (1\,1, \; a\,a\,b\,b\,b)\\ &\vdash (1\,1\,1, \; a\,b\,b\,b)\\ &\vdash (1\,1\,1\,1, \; b\,b\,b)\end{aligned}$$

## ... for example:

**States:** $0, 1, 2$
**Start state:** $0$
**Final states:** $0, 2$

| | | |
|---|---|---|
| $0$ | $a$ | $11$ |
| $1$ | $a$ | $11$ |
| $11$ | $b$ | $2$ |
| $12$ | $b$ | $2$ |

$$
\begin{array}{lll}
(0\,, & a\,a\,a\,b\,b\,b) & \vdash & (1\,1\,, & a\,a\,b\,b\,b) \\
 & & \vdash & (1\,1\,1\,, & a\,b\,b\,b) \\
 & & \vdash & (1\,1\,1\,1\,, & b\,b\,b) \\
 & & \vdash & (1\,1\,2\,, & b\,b)
\end{array}
$$



## ... for example:

**States:** $0, 1, 2$
**Start state:** $0$
**Final states:** $0, 2$

| | | |
|---|---|---|
| $0$ | $a$ | $11$ |
| $1$ | $a$ | $11$ |
| $11$ | $b$ | $2$ |
| $12$ | $b$ | $2$ |

$$
\begin{array}{lll}
(0\,, & a\,a\,a\,b\,b\,b) & \vdash & (1\,1\,, & a\,a\,b\,b\,b) \\
 & & \vdash & (1\,1\,1\,, & a\,b\,b\,b) \\
 & & \vdash & (1\,1\,1\,1\,, & b\,b\,b) \\
 & & \vdash & (1\,1\,2\,, & b\,b) \\
 & & \vdash & (1\,2\,, & b) \\
 & & \vdash & (2\,, & \epsilon)
\end{array}
$$



A computation step is characterized by the relation $\vdash \; \subseteq \; (Q^* \times T^*)^2$ with

$$(\alpha\,\gamma,\; x\,w) \vdash (\alpha\,\gamma',\; w) \quad \text{for} \quad (\gamma,\; x,\; \gamma') \in \delta$$



A computation step is characterized by the relation $\vdash \; \subseteq \; (Q^* \times T^*)^2$ with

$$(\alpha\,\gamma,\; x\,w) \vdash (\alpha\,\gamma',\; w) \quad \text{for} \quad (\gamma,\; x,\; \gamma') \in \delta$$

### Remarks:

- The relation $\vdash$ depends of the pushdown automaton $M$
- The reflexive and transitive closure of $\vdash$ is denoted by $\vdash^*$
- Then, the language accepted by $M$ is

$$\mathcal{L}(M) = \{w \in T^* \mid \exists\, f \in F : (q_0, w) \vdash^* (f, \epsilon)\}$$

### Definition: Deterministic Pushdown Automaton

The pushdown automaton $M$ is deterministic, if every configuration has maximally one successor configuration.

This is exactly the case if for distinct transitions $(\gamma_1, x, \gamma_2)$, $(\gamma_1', x', \gamma_2') \in \delta$ we can assume:
Is $\gamma_1$ a suffix of $\gamma_1'$, then $x \neq x' \ \wedge \ x \neq \epsilon \neq x'$ is valid.



### Definition: Deterministic Pushdown Automaton

The pushdown automaton $M$ is deterministic, if every configuration has maximally one successor configuration.

This is exactly the case if for distinct transitions $(\gamma_1, x, \gamma_2)$, $(\gamma_1', x', \gamma_2') \in \delta$ we can assume:
Is $\gamma_1$ a suffix of $\gamma_1'$, then $x \neq x' \ \wedge \ x \neq \epsilon \neq x'$ is valid.

... for example:

| | | |
|---|---|---|
| 0 | $a$ | 11 |
| 1 | $a$ | 11 |
| 11 | $b$ | 2 |
| 12 | $b$ | 2 |

... this obviously holds



## Pushdown Automata


M. Schützenberger


A. Öttinger

### Theorem:

For each context free grammar $G = (N, T, P, S)$ a pushdown automaton $M$ with $\mathcal{L}(G) = \mathcal{L}(M)$ can be built.

The theorem is so important for us, that we take a look at two constructions for automata, motivated by both of the special derivations:

- $M_G^L$ to build Leftmost derivations
- $M_G^R$ to build reverse Rightmost derivations





# Chapter 3: Top-down Parsing

## Item Pushdown Automaton

**Construction:** Item Pushdown Automaton $M_G^L$

- Reconstruct a Leftmost derivation.
- Expand nonterminals using a rule.
- Verify successively, that the chosen rule matches the input.

$\Longrightarrow$ The states are now Items (= rules with a bullet):

$$[A \rightarrow \alpha \bullet \beta]\,, \qquad A \rightarrow \alpha\,\beta \;\in\; P$$

The bullet marks the spot, how far the rule is already processed



## Item Pushdown Automaton

**Construction:** Item Pushdown Automaton $M_G^L$

- Reconstruct a Leftmost derivation.
- Expand nonterminals using a rule.
- Verify successively, that the chosen rule matches the input.

$\Longrightarrow$ The states are now Items (= rules with a bullet):

$$[A \rightarrow \alpha \bullet \beta]\,, \qquad A \rightarrow \alpha\,\beta \;\in\; P$$

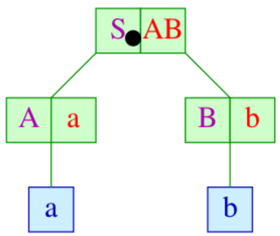
The bullet marks the spot, how far the rule is already processed



## Item Pushdown Automaton – Example

Our example:

$$S \rightarrow A\,B \qquad A \rightarrow a \qquad B \rightarrow b$$

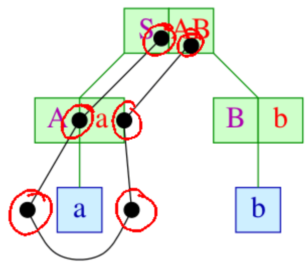





## Item Pushdown Automaton – Example

Our example:

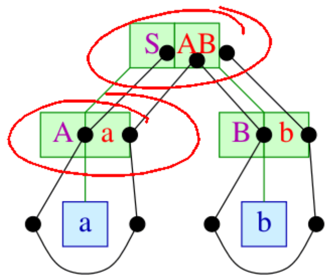
$$S \rightarrow A\,B \qquad A \rightarrow a \qquad B \rightarrow b$$



## Item Pushdown Automaton – Example

### Our example:

$$S \to A\,B \qquad A \to a \qquad B \to b$$






## Item Pushdown Automaton – Example

We add another rule $S' \to S$ for initialising the construction:

**Start state:** $[S' \to \bullet\, S]$
**End state:** $[S' \to S\, \bullet]$
**Transition relations:**

| | | |
|---|---|---|
| $[S' \to \bullet\, S]$ | $\epsilon$ | $[S' \to \bullet\, S]\,[S \to \bullet\, A\,B]$ |
| $[S \to \bullet\, A\,B]$ | $\epsilon$ | $[S \to \bullet\, A\,B]\,[A \to \bullet\, a]$ |
| $[A \to \bullet\, a]$ | $a$ | $[A \to a\, \bullet]$ |
| $[S \to \bullet\, A\,B]\,[A \to a\, \bullet]$ | $\epsilon$ | $[S \to A\, \bullet\, B]$ |
| $[S \to A\, \bullet\, B]$ | $\epsilon$ | $[S \to A\, \bullet\, B]\,[B \to \bullet\, b]$ |
| $[B \to \bullet\, b]$ | $b$ | $[B \to b\, \bullet]$ |
| $[S \to A\, \bullet\, B]\,[B \to b\, \bullet]$ | $\epsilon$ | $[S \to A\,B\, \bullet]$ |
| $[S' \to \bullet\, S]\,[S \to A\,B\, \bullet]$ | $\epsilon$ | $[S' \to S\, \bullet]$ |



## Item Pushdown Automaton

The item pushdown automaton $M_G^L$ has three kinds of transitions:

**Expansions:** $([A \to \alpha \bullet B\,\beta], \epsilon, [A \to \alpha \bullet B\,\beta]\,[B \to \bullet\,\gamma])$ for $A \to \alpha\,B\,\beta,\; B \to \gamma \in P$

**Shifts:** $([A \to \alpha \bullet a\,\beta], a, [A \to \alpha\,a \bullet \beta])$ for $A \to \alpha\,a\,\beta \in P$

**Reduces:** $([A \to \alpha \bullet B\,\beta]\,[B \to \gamma\,\bullet], \epsilon, [A \to \alpha\,B \bullet \beta])$ for $A \to \alpha\,B\,\beta,\; B \to \gamma \in P$

Items of the form: $[A \to \alpha\,\bullet]$ are also called complete

The item pushdown automaton shifts the bullet around the derivation tree ...



## Item Pushdown Automaton – Example

We add another rule $S' \to S$ for initialising the construction:

**Start state:** $[S' \to \bullet\, S]$
**End state:** $[S' \to S\, \bullet]$
**Transition relations:**

| | | |
|---|---|---|
| $[S' \to \bullet\, S]$ | $\epsilon$ | $[S' \to \bullet\, S]\,[S \to \bullet\, A\,B]$ |
| $[S \to \bullet\, A\,B]$ | $\epsilon$ | $[S \to \bullet\, A\,B]\,[A \to \bullet\, a]$ |
| $[A \to \bullet\, a]$ | $a$ | $[A \to a\, \bullet]$ |
| $[S \to \bullet\, A\,B]\,[A \to a\, \bullet]$ | $\epsilon$ | $[S \to A\, \bullet\, B]$ |
| $[S \to A\, \bullet\, B]$ | $\epsilon$ | $[S \to A\, \bullet\, B]\,[B \to \bullet\, b]$ |
| $[B \to \bullet\, b]$ | $b$ | $[B \to b\, \bullet]$ |
| $[S \to A\, \bullet\, B]\,[B \to b\, \bullet]$ | $\epsilon$ | $[S \to A\,B\, \bullet]$ |
| $[S' \to \bullet\, S]\,[S \to A\,B\, \bullet]$ | $\epsilon$ | $[S' \to S\, \bullet]$ |

## Item Pushdown Automaton

### Discussion:

- The expansions of a computation form a leftmost derivation
- Unfortunately, the expansions are chosen nondeterministically
- For proving correctness of the construction, we show that for every Item $[A \to \alpha \bullet B\, \beta]$ the following holds:

$$([A \to \alpha \bullet B\, \beta],\, w) \vdash^{*} ([A \to \alpha\, B \bullet \beta],\, \epsilon) \qquad \text{iff} \qquad B \to^{*} w$$

- LL-Parsing is based on the item pushdown automaton and tries to make the expansions deterministic ...



## Item Pushdown Automaton

**Example:** $S \to \epsilon \;\mid\; a\, S\, b$

The transitions of the according Item Pushdown Automaton:

| | | | |
|---|---|---|---|
| 0 | $[S' \to \bullet\, S]$ | $\epsilon$ | $[S' \to \bullet\, S]\,[S \to \bullet]$ |
| 1 | $[S' \to \bullet\, S]$ | $\epsilon$ | $[S' \to \bullet\, S]\,[S \to \bullet\, a\, S\, b]$ |
| 2 | $[S \to \bullet\, a\, S\, b]$ | $a$ | $[S \to a \bullet S\, b]$ |
| 3 | $[S \to a \bullet S\, b]$ | $\epsilon$ | $[S \to a \bullet S\, b]\,[S \to \bullet]$ |
| 4 | $[S \to a \bullet S\, b]$ | $\epsilon$ | $[S \to a \bullet S\, b]\,[S \to \bullet\, a\, S\, b]$ |
| 5 | $[S \to a \bullet S\, b]\,[S \to \bullet]$ | $\epsilon$ | $[S \to a\, S \bullet b]$ |
| 6 | $[S \to a \bullet S\, b]\,[S \to a\, S\, b \bullet]$ | $\epsilon$ | $[S \to a\, S \bullet b]$ |
| 7 | $[S \to a\, S \bullet b]$ | $b$ | $[S \to a\, S\, b \bullet]$ |
| 8 | $[S' \to \bullet\, S]\,[S \to \bullet]$ | $\epsilon$ | $[S' \to S \bullet]$ |
| 9 | $[S' \to \bullet\, S]\,[S \to a\, S\, b \bullet]$ | $\epsilon$ | $[S' \to S \bullet]$ |



## Topdown Parsing

**Problem:**
Conflicts between the transitions prohibit an implementation of the item pushdown automaton as deterministic pushdown automaton.

**Idea 1: GLL Parsing**
For each conflict, we create a virtual copy of the complete stack and continue deriving in parallel.



## Topdown Parsing

**Problem:**
Conflicts between the transitions prohibit an implementation of the item pushdown automaton as deterministic pushdown automaton.

**Idea 1: GLL Parsing**
For each conflict, we create a virtual copy of the complete stack and continue deriving in parallel.

**Idea 2: Recursive Descent & Backtracking**
Depth-first search for an appropriate derivation.

## Topdown Parsing

**Problem:**
Conflicts between the transitions prohibit an implementation of the item pushdown automaton as deterministic pushdown automaton.

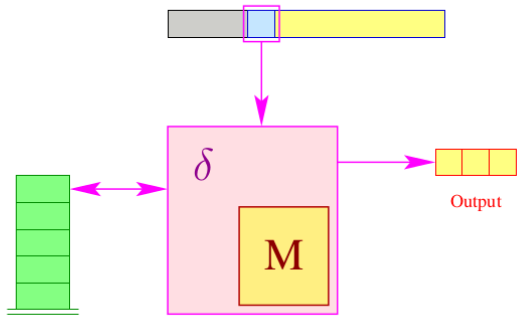
**Idea 1: GLL Parsing**
For each conflict, we create a virtual copy of the complete stack and continue deriving in parallel.

**Idea 2: Recursive Descent & Backtracking**
Depth-first search for an appropriate derivation.

**Idea 3: Recursive Descent & Lookahead**
Conflicts are resolved by considering a lookup of the next input symbol.



## Structure of the $LL(1)$-Parser:




- The parser accesses a frame of length $1$ of the input;
- it corresponds to an item pushdown automaton, essentially;
- table $M[q, w]$ contains the rule of choice.