**Script** generated by TTT

Title: Petter: Compilerbau (18.05.2015)

Date: Mon May 18 14:17:28 CEST 2015

Duration: 91:08 min

Pages: 52

---

### 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 $(\boxed{\gamma_1}, \boxed{x}, \gamma_2)$, $(\boxed{\gamma_1'}, \boxed{x'}, \gamma_2') \in \delta$ 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 $\boxed{\mathcal{L}(G) = \mathcal{L}(M)}$ 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:

- $\boxed{M_G^L}$ to build Leftmost derivations
- $\boxed{M_G^R}$ to build reverse Rightmost derivations

## 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):

$$\boxed{[A \to \boxed{\alpha} \bullet \boxed{\beta}]}, \qquad A \to \alpha\,\beta \;\in\; P$$

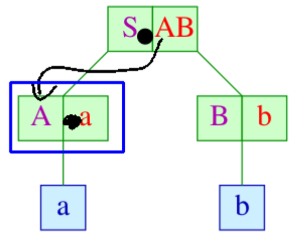
The bullet marks the spot, how far the rule is already processed



## Item Pushdown Automaton – Example

Our example:

$$S \to A\,B \qquad \boxed{A \to a} \qquad B \to b$$

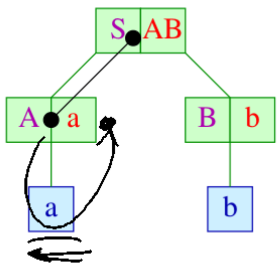





## Item Pushdown Automaton – Example

Our example:

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



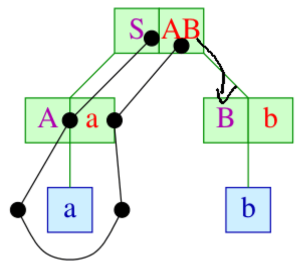

B



## Item Pushdown Automaton – Example

Our example:

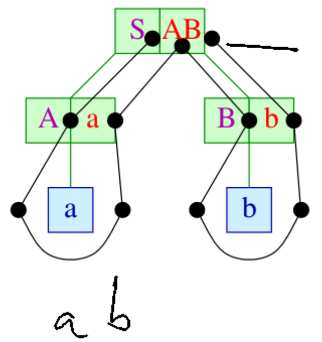
$$S \to A\,B \qquad A \to a \qquad B \to b$$



## 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:** $([A \to \alpha \bullet B\,\beta], \boxed{\epsilon}, \boxed{[A \to \alpha \bullet B\,\beta]}\,\boxed{[B \to \bullet\, \gamma]})$ 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

### Discussion:

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

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

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

## Item Pushdown Automaton

**Example:** $S \to \epsilon \quad | \quad a\,S\,b$

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.



## 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.

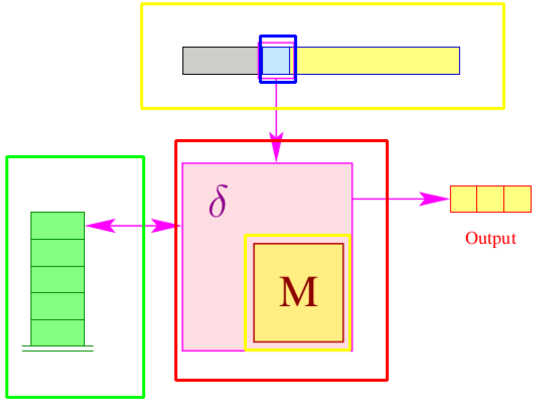
**Idea 1: GLL Parsing**
For each conflict, we create a virtual copy of the complete stack and continue deriving in parallel.

**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.



## Topdown Parsing

**Idea:**

- Emanate from the item pushdown automaton
- Consider the next input symbol to determine the appropriate rule for the next expansion
- A grammar is called $LL(1)$ if a unique choice is always possible



## Topdown Parsing

**Idea:**

- Emanate from the item pushdown automaton
- Consider the next input symbol to determine the appropriate rule for the next expansion
- A grammar is called $LL(1)$ if a unique choice is always possible


Philip Lewis


Richard Stearns

**Definition:**
A reduced grammar is called $LL(1)$,
if for each two distinct rules $A \to \alpha$, $A \to \alpha' \in P$ and each derivation $S \to_L^* u A \beta$ with $u \in T^*$ the following is valid:

$$\text{First}_1(\alpha\beta) \cap \text{First}_1(\alpha'\beta) = \emptyset$$

## Topdown Parsing

### Example 1:

$$
\begin{array}{rcl}
S & \rightarrow & \boxed{\text{if}}\ (\ E\ )\ S\ \text{else}\ S \quad | \\
  &             & \boxed{\text{while}}\ (\ E\ )\ S \quad | \\
  &             & \boxed{E}\ ; \\
E & \rightarrow & \text{id}
\end{array}
$$

if
while
id

is $LL(1)$, since $\text{First}_1(E) = \{\text{id}\}$



## Topdown Parsing

### Example 1:

$$
\begin{array}{rcl}
S & \rightarrow & \text{if}\ (\ E\ )\ S\ \text{else}\ S \quad | \\
  &             & \text{while}\ (\ E\ )\ S \quad | \\
  &             & E\ ; \\
E & \rightarrow & \text{id}
\end{array}
$$

is $LL(1)$, since $\text{First}_1(E) = \{\text{id}\}$

### Example 2:

$$
\begin{array}{rcl}
S & \rightarrow & \text{if}\ (\ E\ )\ S\ \text{else}\ S \quad | \\
  &             & \text{if}\ (\ E\ )\ S \quad | \\
  &             & \text{while}\ (\ E\ )\ S \quad | \\
  &             & E\ ; \\
E & \rightarrow & \text{id}
\end{array}
$$

if (id) if (id) S
if (id) if (id) S

... is not $LL(k)$ for any $k > 0$.



## Lookahead Sets

**Definition: $\text{First}_1$-Sets**

For a set $L \subseteq T^*$ we define:

$$\text{First}_1(L) \;=\; \{\epsilon \mid \epsilon \in L\} \cup \{u \in T \mid \exists v \in T^* :\; uv \in L\}$$

Example: $\boxed{S \rightarrow \epsilon \quad | \quad a\,S\,b}$

| $\text{First}_1(S)$ |
|---|
| $\epsilon$ |
| $a\,b$ |
| $a\,a\,b\,b$ |
| $a\,a\,a\,b\,b\,b$ |
| $\dots$ |

$= \{\varepsilon, a\}$



## Lookahead Sets

**Definition: $\text{First}_1$-Sets**

For a set $L \subseteq T^*$ we define:

$$\text{First}_1(L) \;=\; \{\epsilon \mid \epsilon \in L\} \cup \{u \in T \mid \exists v \in T^* :\; uv \in L\}$$

Example: $S \rightarrow \epsilon \quad | \quad a\,S\,b$

| $\text{First}_1(S)$ |
|---|
| $\epsilon$ |
| $a\,b$ |
| $a\,a\,b\,b$ |
| $a\,a\,a\,b\,b\,b$ |
| $\dots$ |

$\equiv$ the yield's prefix of length 1

## Lookahead Sets

Arithmetics:
$\text{First}_1(\_)$ is compatible with union and concatenation:

$$\begin{array}{lll}
\text{First}_1(\emptyset) & = & \emptyset \\
\text{First}_1(L_1 \cup L_2) & = & \text{First}_1(L_1) \cup \text{First}_1(L_2) \\
\text{First}_1(L_1 \cdot L_2) & = & \text{First}_1(\text{First}_1(L_1) \cdot \text{First}_1(L_2)) \\
& := & \text{First}_1(L_1) \odot \text{First}_1(L_2)
\end{array}$$

$\odot$ being $1$ – concatenation



## Lookahead Sets

Arithmetics:
$\text{First}_1(\_)$ is compatible with union and concatenation:

$$\begin{array}{lll}
\text{First}_1(\emptyset) & = & \emptyset \\
\text{First}_1(L_1 \cup L_2) & = & \text{First}_1(L_1) \cup \text{First}_1(L_2) \\
\text{First}_1(L_1 \cdot L_2) & = & \boxed{\text{First}_1(\text{First}_1(L_1) \cdot \text{First}_1(L_2))} \\
& := & \text{First}_1(L_1) \odot \text{First}_1(L_2)
\end{array}$$

$\odot$ being $1$ – concatenation

**Definition: 1-concatenation**

Let $L_1, L_2 \subseteq T \cup \{\epsilon\}$ with $L_1 \neq \emptyset \neq L_2$. Then:

$$L_1 \odot L_2 = \begin{cases} L_1 & \text{if } \epsilon \notin L_1 \\ (L_1 \backslash \{\epsilon\}) \cup L_2 & \text{otherwise} \end{cases}$$

If all rules of $G$ are productive, then all sets $\text{First}_1(A)$ are non-empty.



## Lookahead Sets

For $\alpha \in (N \cup T)^*$ we are interested in the set:

$$\text{First}_1(\alpha) = \text{First}_1(\{w \in T^* \mid \alpha \to^* w\})$$

Idea: Treat $\epsilon$ separately: $\text{First}_1(A) = F_\epsilon(A) \cup \{\epsilon \mid A \to^* \epsilon\}$

- Let $\text{empty}(X) = \text{true}$ iff $X \to^* \epsilon$ .
- $F_\epsilon(X_1 \dots X_m) = \bigcup_{i=1}^{j} F_\epsilon(X_i)$ if $\bigwedge_{i=1}^{j-1} \text{empty}(X_i)$



## Lookahead Sets

For $\alpha \in (N \cup T)^*$ we are interested in the set:

$$\text{First}_1(\alpha) = \text{First}_1(\{w \in T^* \mid \alpha \to^* w\})$$

Idea: Treat $\epsilon$ separately: $\text{First}_1(A) = F_\epsilon(A) \cup \{\epsilon \mid A \to^* \epsilon\}$

- Let $\text{empty}(X) = \text{true}$ iff $X \to^* \epsilon$ .
- $F_\epsilon(X_1 \dots X_m) = \bigcup_{i=1}^{j} F_\epsilon(X_i)$ if $\bigwedge_{i=1}^{j-1} \text{empty}(X_i)$

We characterize the $\epsilon$-free $\text{First}_1$-sets with an inequality system:

$$\begin{array}{llll}
F_\epsilon(a) & = & \{a\} & \text{if } a \in T \\
F_\epsilon(A) & \supseteq & F_\epsilon(X_j) & \text{if } A \to X_1 \dots X_m \in P, \\
& & & \bigwedge_{i=1}^{j-1} \text{empty}(X_i)
\end{array}$$

## Lookahead Sets

for example...

$$\begin{array}{lcll} E & \to & E+T & \mid \; T \\ T & \to & T*F & \mid \; F \\ F & \to & (\; E\;) & \mid \; \text{name} \quad \mid \; \text{int} \end{array}$$

with $\text{empty}(E) = \text{empty}(T) = \text{empty}(F) = \text{false}$



## Lookahead Sets

for example...

$$\begin{array}{lcll} E & \to & E+T & \mid \; T \\ T & \to & T*F & \mid \; F \\ F & \to & (\; E\;) & \mid \; \text{name} \quad \mid \; \text{int} \end{array}$$

with $\text{empty}(E) = \text{empty}(T) = \text{empty}(F) = \text{false}$

... we obtain:

$$\begin{array}{lcl@{\qquad}lcl} F_\epsilon(S') & \supseteq & F_\epsilon(E) & F_\epsilon(E) & \supseteq & F_\epsilon(E) \\ F_\epsilon(E) & \supseteq & F_\epsilon(T) & F_\epsilon(T) & \supseteq & F_\epsilon(T) \\ F_\epsilon(T) & \supseteq & F_\epsilon(F) & F_\epsilon(F) & \supseteq & \{\,(\,,\text{name},\text{int}\} \end{array}$$



## Lookahead Sets

For $\alpha \in (N \cup T)^*$ we are interested in the set:

$$\text{First}_1(\alpha) \;=\; \text{First}_1(\{w \in T^* \mid \alpha \to^* w\})$$

Idea: Treat $\epsilon$ separately: $\text{First}_1(A) = F_\epsilon(A) \cup \{\epsilon \mid A \to^* \epsilon\}$

- Let $\text{empty}(X) = \text{true}$ iff $X \to^* \epsilon$ .
- $F_\epsilon(X_1 \dots X_m) = \bigcup_{i=1}^{j} F_\epsilon(X_i)$ if $\bigwedge_{i=1}^{j-1} \text{empty}(X_i)$

We characterize the $\epsilon$-free $\text{First}_1$-sets with an inequality system:

$$\begin{array}{lcll} F_\epsilon(a) & = & \{a\} & \text{if} \quad a \in T \\ F_\epsilon(A) & \supseteq & F_\epsilon(X_j) & \text{if} \quad A \to X_1 \dots X_m \in P, \\ & & & \quad \bigwedge_{i=1}^{j-1} \text{empty}(X_i) \end{array}$$



## Lookahead Sets

for example...

$$\begin{array}{lcll} E & \to & E+T & \mid \; T \\ T & \to & T*F & \mid \; F \\ F & \to & (\; E\;) & \mid \; \text{name} \quad \mid \; \text{int} \end{array}$$

with $\text{empty}(E) = \text{empty}(T) = \text{empty}(F) = \text{false}$

... we obtain:

$$\begin{array}{lcl@{\qquad}lcl} F_\epsilon(S') & \supseteq & F_\epsilon(E) & F_\epsilon(E) & \supseteq & \boxed{F_\epsilon(E)} \\ F_\epsilon(E) & \supseteq & F_\epsilon(T) & F_\epsilon(T) & \supseteq & F_\epsilon(T) \\ F_\epsilon(T) & \supseteq & F_\epsilon(F) & F_\epsilon(F) & \supseteq & \boxed{\{\,(\,,\text{name},\text{int}\}} \end{array}$$

## Fast Computation of Lookahead Sets

### Observation:

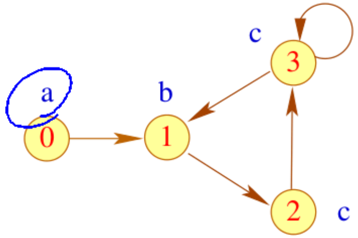
- The form of each inequality of these systems is:

$$x \sqsupseteq y \qquad \text{resp.} \qquad x \sqsupseteq d$$

for variables $x, y$ und $d \in \mathbb{D}$.
- Such systems are called pure unification problems
- Such problems can be solved in linear space/time.

### for example: $\mathbb{D} = 2^{\{a,b,c\}}$

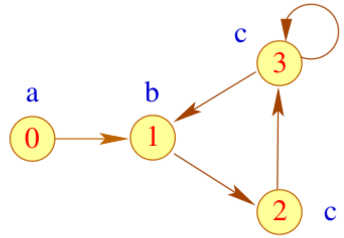
$$\begin{array}{lll} x_0 \supseteq \{a\} & & \\ x_1 \supseteq \{b\} & x_1 \supseteq x_0 & x_1 \supseteq x_3 \\ x_2 \supseteq \{c\} & x_2 \supseteq x_1 & \\ x_3 \supseteq \{c\} & x_3 \supseteq x_2 & x_3 \supseteq x_3 \end{array}$$






## Fast Computation of Lookahead Sets


Frank DeRemer & Tom Pennello



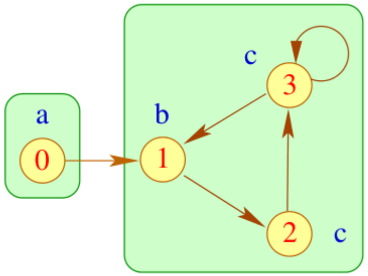

### Proceeding:

- Create the Variable Dependency Graph for the inequality system.



## Fast Computation of Lookahead Sets


Frank DeRemer & Tom Pennello



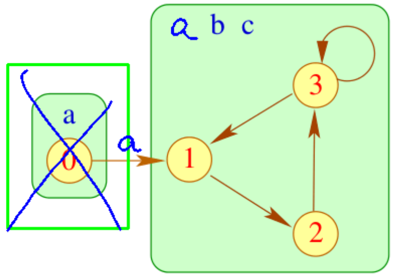

### Proceeding:

- Create the Variable Dependency Graph for the inequality system.
- Whithin a Strongly Connected Component ($\rightarrow$ Tarjan) all variables have the same value



## Fast Computation of Lookahead Sets


Frank DeRemer & Tom Pennello

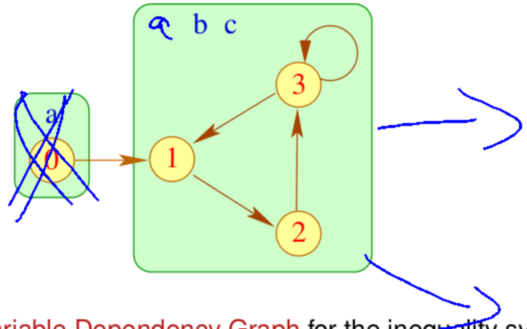



### Proceeding:

- Create the Variable Dependency Graph for the inequality system.
- Whithin a Strongly Connected Component ($\rightarrow$ Tarjan) all variables have the same value
- Is there no ingoing edge for an SCC, its value is computed via the smallest upper bound of all values within the SCC

## Fast Computation of Lookahead Sets

Frank DeRemer
& Tom Pennello



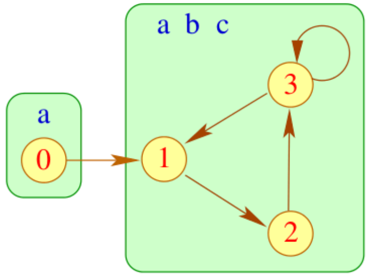

### Proceeding:

- Create the Variable Dependency Graph for the inequality system.
- Whithin a Strongly Connected Component ($\rightarrow$ Tarjan) all variables have the same value
- Is there no ingoing edge for an SCC, its value is computed via the smallest upper bound of all values within the SCC



## Fast Computation of Lookahead Sets

Frank DeRemer
& Tom Pennello




### Proceeding:

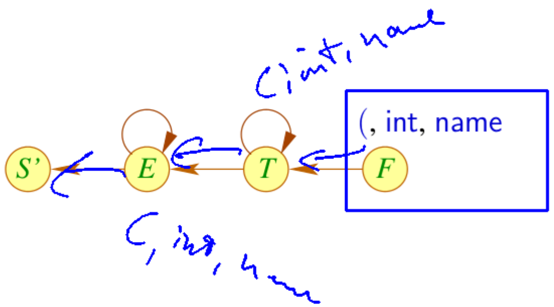
- Create the Variable Dependency Graph for the inequality system.
- Whithin a Strongly Connected Component ($\rightarrow$ Tarjan) all variables have the same value
- Is there no ingoing edge for an SCC, its value is computed via the smallest upper bound of all values within the SCC
- In case of ingoing edges, their values are also to be considered for the upper bound



## Fast Computation of Lookahead Sets

... for our example grammar:

First$_1$ :






## Item Pushdown Automaton as LL(1)-Parser

back to the example: $S \rightarrow \epsilon \quad | \quad a\,S\,b$

The transitions in the according Item Pushdown Automaton:

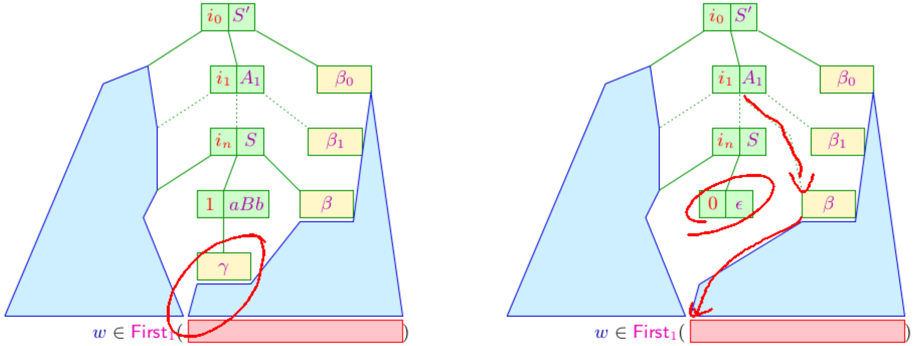
| | | | |
|---|---|---|---|
| 0 | $[S' \rightarrow \bullet\, S]$ | $\epsilon$ | $[S' \rightarrow \bullet\, S]\,[S \rightarrow \bullet]$ |
| 1 | $[S' \rightarrow \bullet\, S]$ | $\epsilon$ | $[S' \rightarrow \bullet\, S]\,[S \rightarrow \bullet\, a\,S\,b]$ |
| 2 | $[S \rightarrow \bullet\, a\,S\,b]$ |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

Conflicts arise between transations $(0, 1)$ or $(3, 4)$ resp..

## Item Pushdown Automaton as LL(1)-Parser

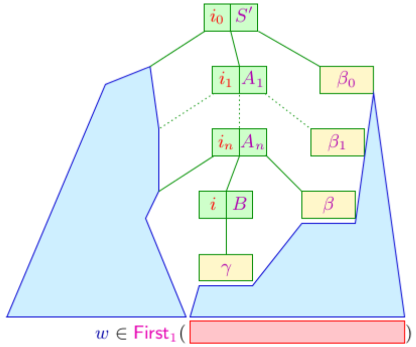
... in detail: $S \to \epsilon^{0} \quad | \quad a\, S\, b^{1}$

| $\mathsf{First}_1(\mathsf{input})$ | $\epsilon$ | $a$ | $b$ |
|---|---|---|---|
| $S$ | ? | ? | ? |





## Item Pushdown Automaton as LL(1)-Parser

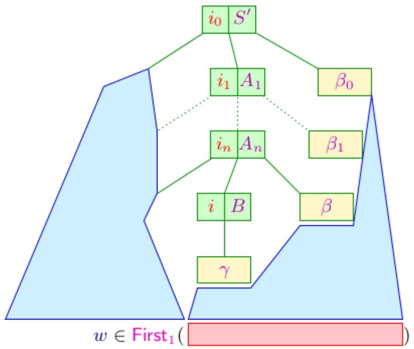


$w \in \mathsf{First}_1(\mathsf{First}_1(\gamma) \odot_1 \mathsf{First}_1(\beta) \odot_1 \ldots \odot_1 \mathsf{First}_1(\beta_0))$

$w \in \mathsf{First}_1(\gamma) \odot_1 \mathsf{Follow}_1(B)$



## Item Pushdown Automaton as LL(1)-Parser



$w \in \mathsf{First}_1(\mathsf{First}_1(\gamma) \odot_1 \mathsf{First}_1(\beta) \odot_1 \ldots \odot_1 \mathsf{First}_1(\beta_0))$

$w \in \mathsf{First}_1(\gamma) \odot_1 \mathsf{Follow}_1(B)$

Inequality system for $\mathsf{Follow}_1(B) = \mathsf{First}_1(\beta) \odot_1 \ldots \odot_1 \mathsf{First}_1(\beta_0)$

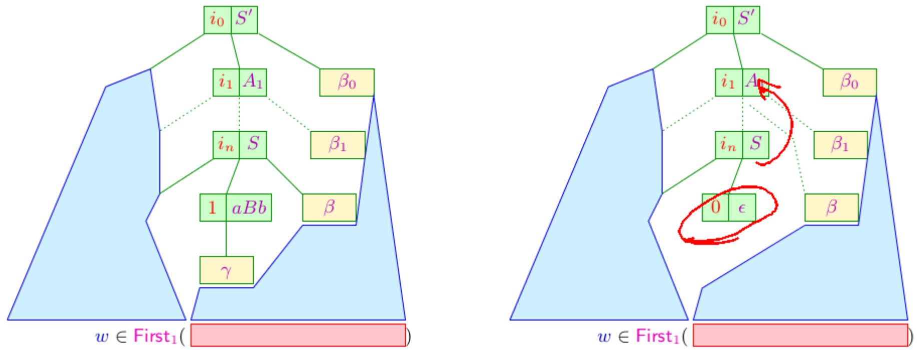
$$\begin{array}{lllll}
\mathsf{Follow}_1(S) & \supseteq & \{\epsilon\} & & \\
\mathsf{Follow}_1(B) & \supseteq & F_\epsilon(X_j) & \text{if} & A \to \alpha\, B\, X_1 \ldots X_m \in P, \\
 & & & & \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_{j-1}) \\
\mathsf{Follow}_1(B) & \supseteq & \mathsf{Follow}_1(A) & \text{if} & A \to \alpha\, B\, X_1 \ldots X_m \in P, \\
 & & & & \mathsf{empty}(X_1) \wedge \ldots \wedge \mathsf{empty}(X_m)
\end{array}$$



## Item Pushdown Automaton as LL(1)-Parser

... in detail: $S \to \epsilon^{0} \quad | \quad a\, S\, b^{1}$

| $\mathsf{First}_1(\mathsf{input})$ | $\epsilon$ | $a$ | $b$ |
|---|---|---|---|
| $S$ | ? | ? | ? |



## Item Pushdown Automaton as LL(1)-Parser

Is $G$ an $LL(1)$-grammar, we can index a lookahead-table with items and nonterminals:

**LL(1)-Lookahead Table**

We set $M[B,\, w] \;=\; i$ with $B \to \gamma^{\,i}$ exactly if

- $S' \to_L^* u\, B\, \beta$
- $w \in \text{First}_1(\gamma) \odot_1 \text{Follow}_1(\beta)$

... for example: $\boxed{S \to \epsilon^{\,0} \quad | \quad a\, S\, b^{\,1}}$



## Item Pushdown Automaton as LL(1)-Parser

Is $G$ an $LL(1)$-grammar, we can index a lookahead-table with items and nonterminals:

**LL(1)-Lookahead Table**

We set $M[B,\, w] \;=\; i$ with $B \to \gamma^{\,i}$ exactly if

- $S' \to_L^* u\, B\, \beta$
- $w \in \text{First}_1(\gamma) \odot_1 \text{Follow}_1(\beta)$

... for example: $\boxed{S \to \epsilon^{\,0}} \quad | \quad \boxed{a}\, S\, b^{\,1}$

$$\text{First}_1(S) = \{\epsilon, a\}$$



## Item Pushdown Automaton as LL(1)-Parser

Is $G$ an $LL(1)$-grammar, we can index a lookahead-table with items and nonterminals:

**LL(1)-Lookahead Table**

We set $M[B,\, w] \;=\; i$ with $B \to \gamma^{\,i}$ exactly if

- $S' \to_L^* u\, B\, \beta$
- $w \in \text{First}_1(\gamma) \odot_1 \text{Follow}_1(\beta)$

... for example: $S \to \epsilon^{\,0} \quad | \quad a\, S\, \boxed{b}^{\,1}$

$$\text{First}_1(S) = \{\epsilon, a\} \qquad \text{Follow}_1(S) = \{\boxed{b}, \boxed{\epsilon}\}$$



## Item Pushdown Automaton as LL(1)-Parser

Is $G$ an $LL(1)$-grammar, we can index a lookahead-table with items and nonterminals:

**LL(1)-Lookahead Table**

We set $M[B,\, w] \;=\; i$ with $B \to \gamma^{\,i}$ exactly if

- $S' \to_L^* u\, B\, \beta$
- $w \in \text{First}_1(\gamma) \odot_1 \text{Follow}_1(\beta)$

... for example: $S \to \epsilon^{\,0} \quad | \quad a\, S\, b^{\,1}$

$$\text{First}_1(S) = \{\epsilon, a\} \qquad \text{Follow}_1(S) = \{b, \epsilon\}$$

$$\begin{array}{lrcl} S\text{-rule } 0: & \text{First}_1(\epsilon) & \odot_1 & \text{Follow}_1(S) = \{b, \epsilon\} \\ S\text{-rule } 1: & \text{First}_1(aSb) & \odot_1 & \text{Follow}_1(S) = \{a\} \end{array}$$

## Item Pushdown Automaton as LL(1)-Parser

Is $G$ an $LL(1)$-grammar, we can index a lookahead-table with items and nonterminals:

**LL(1)-Lookahead Table**

We set $M[B,\ w] \ = \ i$ with $B \to \gamma^{\,i}$ exactly if

- $S' \to_L^* u\, B\, \beta$
- $w \in \text{First}_1(\gamma) \odot_1 \text{Follow}_1(\beta)$

... for example: $S \to \epsilon^{\,0} \quad | \quad a\, S\, b^{\,1}$

$\text{First}_1(S) = \{\epsilon, a\} \quad \text{Follow}_1(S) = \{b, \epsilon\}$

$S$-rule $0$ : $\text{First}_1(\epsilon) \odot_1 \text{Follow}_1(S) = \{b, \epsilon\}$
$S$-rule $1$ : $\text{First}_1(aSb) \odot_1 \text{Follow}_1(S) = \{a\}$

| | $\epsilon$ | $a$ | $b$ |
|---|---|---|---|
| $S$ | 0 | 1 | 0 |



## Item Pushdown Automaton as LL(1)-Parser

For example: $S \to \epsilon^{\,0} \quad | \quad a\, S\, b^{\,1}$

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

Lookahead table:

| | $\epsilon$ | $a$ | $b$ |
|---|---|---|---|
| $S$ | 0 | 1 | 0 |



## Topdown-Parsing

### Discussion

- A practical implementation of an $LL(1)$-parser via recursive Descent is a straight-forward idea
- However, only a subset of the deterministic contextfree languages can be parsed this way.



## Topdown-Parsing

### Discussion

- A practical implementation of an $LL(1)$-parser via recursive Descent is a straight-forward idea
- However, only a subset of the deterministic contextfree languages can be parsed this way.
- Solution: Going from $LL(1)$ to $LL(k)$
- The size of the occuring sets is rapidly increasing with larger $k$
- Unfortunately, even $LL(k)$ parsers are not sufficient to accept all deterministic contextfree languages.
- In practical systems, this often motivates the implementation of $k = 1$ only ...

# Chapter 4:
# Bottom-up Analysis