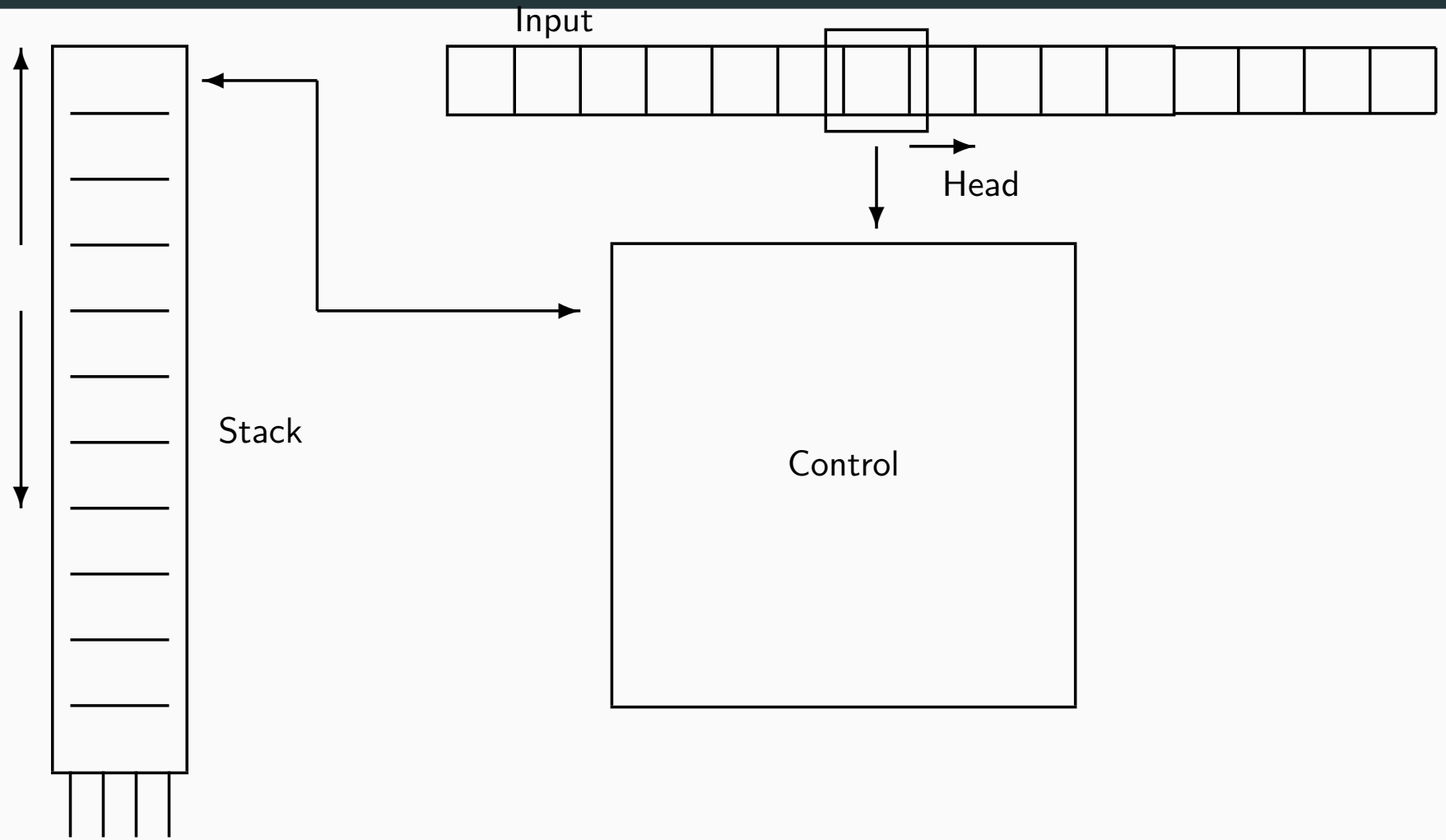
Pushdown Automata and Parser

Sebastian Hack
(based on slides by Reinhard Wilhelm and Mooly Sagiv)

http://compilers.cs.uni-saarland.de

Compiler Construction Core Course 2017
Saarland University

Pushdown Automata



Memory unboundedly extensible at one end,

grows (by push), shrinks (by pop), test for emptiness.

Example Automaton

- Accepted language $L = \{a^i\, b^i \mid i \geq 0\}$
- Context Free Grammar $S \to aSb|\varepsilon$
- Pushdown automaton (TOS = top of stack)

TOS	**input**			
	a	b	ε	\$
(0)	$\binom{1}{0}$	(3)	(3)	(4)
(1)	$\binom{1}{1}$	(2)	(3)	(3)
$\binom{2}{1}$	(3)	(2)	(3)	(3)
$\binom{2}{0}$	(3)	(3)	(3)	(4)

state 0: Initial state,

state 1: reading a's

state 2: reading b's

state 3: error state

state 4: final state.

Pushdown Automaton (PDA) Definition

A tuple $P = (V, Q, \Delta, q_0, F)$ where:

- V — input alphabet
- Q — finite set of states (stack symbols)
- $q_0 \in Q$ — initial state
- $F \subseteq Q$ — final states
- $\Delta \subseteq (Q^+ \times (V \cup \{\varepsilon\})) \times Q^*$
- Alternatively: $\delta\colon (Q^+ \times (V \cup \{\varepsilon\})) \to 2^{Q^*}$ where δ is a partial function

The Language Accepted by a PDA

- PDA $P = (V, Q, \Delta, q_0, F)$
- For $\gamma \in Q^+$, $w \in V^*$, (γ, w) is a configuration
- The binary relation step on configurations is defined by: $(\gamma, aw) \vdash_P (\gamma', w)$ if there exists γ_1 such that
  - $\gamma \equiv \gamma_1\, \gamma_2$
  - $\gamma' \equiv \gamma_1\, \gamma_3$
  - $(\gamma_2, a, \gamma_3) \in \Delta$
- $\vdash_P^*$ is the reflexive transitive closure of $\vdash_P$
- The language accepted by P

$$L(P) = \{w \in V^* \mid \exists q_f \in F : (q_0, w) \vdash_M^* (q_f, \varepsilon)\}$$

Deterministic Pushdown Automaton

- For every $a \in V$, $(\gamma_1, a, \gamma_2), (\gamma_1', a, \gamma_2') \in \Delta$ such that γ_1' is a suffix of γ_1 implies $\gamma_1 = \gamma_1'$ and $\gamma_2 = \gamma_2'$
- There exist no $(\gamma_1, \varepsilon, \gamma_2), (\gamma_1', a, \gamma_2') \in \Delta$ such that $a \in V \cup \{\varepsilon\}$ and γ_1' is a suffix of γ_1 or vice versa.

Theoretical Results

Theorem

For every context free grammar G there exists a non-deterministic pushdown automaton P such that $L(G) = L(P)$

Proof: A PDA is given which emulates the original grammar.

Context Free Items

- A (context–free) item is a triple (A, α, β) where $A \to \alpha\beta \in P$
- An item (A, α, β) is denoted by $[A \to \alpha.\beta]$
- Interpretation:

  > *"In an attempt to recognize a word for A, a word for α has already been recognized"*

  α — history of the item $[A \to \alpha.\beta]$
- $[A \to \alpha.]$ — A complete item
- IT_G — The set of items of G
- $hist([A_1 \to \alpha_1.\beta_1][A_2 \to \alpha_2.\beta_2] \dots [A_n \to \alpha_n.\beta_n]) = \alpha_1\alpha_2 \dots \alpha_n$

The Item Pushdown Automaton

- A context-free grammar $G = (V_N, V_T, P, S)$
- Extended grammar: Add non-term S', and production $S' \to S$
- $P_G = (V_T, IT_G, \delta, [S' \to .S], \{[S' \to S.]\})$
- Control δ

TOS	**input**	**new TOS**	**comment**
$[X \to \beta.Y\gamma]$	ε	$[X \to \beta.Y\gamma][Y \to .\alpha]$	$Y \to \alpha \in P$ "expand"
$[X \to \beta.a\gamma]$	a	$[X \to \beta a.\gamma]$	"shift"
$[X \to \beta.Y\gamma][Y \to \alpha.]$	ε	$[X \to \beta Y.\gamma]$	"reduce"

The Item Pushdown Automaton

- A context-free grammar $G = (V_N, V_T, P, S)$
- Extended grammar: Add non-term S', and production $S' \to S$
- $P_G = (V_T, IT_G, \delta, [S' \to .S], \{[S' \to S.]\})$
- Control δ

TOS	**input**	**new TOS**	**comment**
$[X \to \beta.Y\gamma]$	ε	$[X \to \beta.Y\gamma][Y \to .\alpha]$	$Y \to \alpha \in P$ "expand"
$[X \to \beta.a\gamma]$	a	$[X \to \beta a.\gamma]$	"shift"
$[X \to \beta.Y\gamma][Y \to \alpha.]$	ε	$[X \to \beta Y.\gamma]$	"reduce"

Source of nondeterminism: expansion transitions: there may be several productions for Y

Example

$$P = \{1 : S' \to S, \quad 2 : S \to \varepsilon, \quad 3 : S \to aSb\}$$

TOS	input	new TOS	comment
$[S' \to .S]$	ε	$[S' \to .S][S \to .]$	$e_{1,2}$
$[S' \to .S]$	ε	$[S' \to .S][S \to .aSb]$	$e_{1,3}$
$[S \to a.Sb]$	ε	$[S \to a.Sb][S \to .]$	$e_{2,2}$
$[S \to a.Sb]$	ε	$[S \to a.Sb][S \to .aSb]$	$e_{2,3}$
$[S \to .aSb]$	a	$[S \to a.Sb]$	s_1
$[S \to aS.b]$	b	$[S \to aSb.]$	s_2
$[S' \to .S][S \to .]$	ε	$[S' \to S.]$	r_1
$[S' \to .S][S \to aSb.]$	ε	$[S' \to S.]$	r_2
$[S \to a.Sb][S \to .]$	ε	$[S \to aS.b]$	r_3
$[S \to a.Sb][S \to aSb.]$	ε	$[S \to aS.b]$	r_4

Automaton for the Expression Grammar G_0

TOS	Input	New TOS
$[S \rightarrow .E]$	ε	$[S \rightarrow .E][E \rightarrow .E + T]$
$[S \rightarrow .E]$	ε	$[S \rightarrow .E][E \rightarrow .T]$
$[E \rightarrow .E + T]$	ε	$[E \rightarrow .E + T][E \rightarrow .E + T]$
$[E \rightarrow .E + T]$	ε	$[E \rightarrow .E + T][E \rightarrow .T]$
$[F \rightarrow (.E)]$	ε	$[F \rightarrow (.E)][E \rightarrow .E + T]$
$[F \rightarrow (.E)]$	ε	$[F \rightarrow (.E)][E \rightarrow .T]$
$[E \rightarrow .T]$	ε	$[E \rightarrow .T][T \rightarrow .T * F]$
$[E \rightarrow .T]$	ε	$[E \rightarrow .T][T \rightarrow .F]$
$[T \rightarrow .T * F]$	ε	$[T \rightarrow .T * F][T \rightarrow .T * F]$
$[T \rightarrow .T * F]$	ε	$[T \rightarrow .T * F][T \rightarrow .F]$
$[E \rightarrow E + .T]$	ε	$[E \rightarrow E + .T][T \rightarrow .T * F]$
$[E \rightarrow E + .T]$	ε	$[E \rightarrow E + .T][T \rightarrow .F]$
$[T \rightarrow .F]$	ε	$[T \rightarrow .F][F \rightarrow .(E)]$
$[T \rightarrow .F]$	ε	$[T \rightarrow .F][F \rightarrow .\mathbf{id}]$
$[T \rightarrow T * .F]$	ε	$[T \rightarrow T * .F][F \rightarrow .(E)]$
$[T \rightarrow T * .F]$	ε	$[T \rightarrow T * .F][F \rightarrow .\mathbf{id}]$

TOS	Input	New TOS
$[F \rightarrow .(E)]$	$($	$[F \rightarrow (.E)]$
$[F \rightarrow .\mathbf{id}]$	$\mathbf{id}$	$[F \rightarrow \mathbf{id}.]$
$[F \rightarrow (E.)]$	$)$	$[E \rightarrow (E).]$
$[E \rightarrow E. + T]$	$+$	$[E \rightarrow E + .T]$
$[T \rightarrow T. * F]$	$*$	$[T \rightarrow T * .F]$
$[T \rightarrow .F][F \rightarrow \mathbf{id}.]$	ε	$[T \rightarrow F.]$
$[T \rightarrow T * .F][F \rightarrow \mathbf{id}.]$	ε	$[T \rightarrow T * F.]$
$[T \rightarrow .F][F \rightarrow (E).]$	ε	$[T \rightarrow F.]$
$[T \rightarrow T * .F][F \rightarrow (E).]$	ε	$[T \rightarrow T * F.]$
$[T \rightarrow .T * F][T \rightarrow F.]$	ε	$[T \rightarrow T. * F]$
$[E \rightarrow .T][T \rightarrow F.]$	ε	$[E \rightarrow T.]$
$[E \rightarrow E + .T][T \rightarrow F.]$	ε	$[E \rightarrow E + T.]$
$[E \rightarrow E + .T][T \rightarrow T * F.]$	ε	$[E \rightarrow E + T.]$
$[T \rightarrow .T * F][T \rightarrow T * F.]$	ε	$[T \rightarrow T. * F]$
$[E \rightarrow .T][T \rightarrow T * F.]$	ε	$[E \rightarrow T.]$
$[F \rightarrow (.E)][E \rightarrow T.]$	ε	$[F \rightarrow (E.)]$
$[F \rightarrow (.E)][E \rightarrow E + T.]$	ε	$[F \rightarrow (E.)]$
$[E \rightarrow .E + T][E \rightarrow T.]$	ε	$[E \rightarrow E. + T]$
$[E \rightarrow .E + T][E \rightarrow E + T.]$	ε	$[E \rightarrow E. + T]$
$[S \rightarrow .E][E \rightarrow T.]$	ε	$[S \rightarrow E.]$
$[S \rightarrow .E][E \rightarrow E + T.]$	ε	$[S \rightarrow E.]$

Stack when accepting $id + id * id$:	Remaining Input
$[S \rightarrow .E]$	**id** $+$ **id** $*$ **id**
$[S \rightarrow .E][E \rightarrow .E + T]$	**id** $+$ **id** $*$ **id**
$[S \rightarrow .E][E \rightarrow .E + T][E \rightarrow .T]$	**id** $+$ **id** $*$ **id**
$[S \rightarrow .E][E \rightarrow .E + T][E \rightarrow .T][T \rightarrow .F]$	**id** $+$ **id** $*$ **id**
$[S \rightarrow .E][E \rightarrow .E + T][E \rightarrow .T][T \rightarrow .F][F \rightarrow .\mathbf{id}]$	**id** $+$ **id** $*$ **id**
$[S \rightarrow .E][E \rightarrow .E + T][E \rightarrow .T][T \rightarrow .F][F \rightarrow \mathbf{id}.]$	$+$**id** $*$ **id**
$[S \rightarrow .E][E \rightarrow .E + T][E \rightarrow .T][T \rightarrow F.]$	$+$**id** $*$ **id**
$[S \rightarrow .E][E \rightarrow .E + T][E \rightarrow T.]$	$+$**id** $*$ **id**
$[S \rightarrow .E][E \rightarrow E. + T]$	$+$**id** $*$ **id**
$[S \rightarrow .E][E \rightarrow E + .T]$	**id** $*$ **id**
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow .T * F]$	**id** $*$ **id**
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow .T * F][T \rightarrow .F]$	**id** $*$ **id**
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow .T * F][T \rightarrow .F][F \rightarrow .\mathbf{id}]$	**id** $*$ **id**
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow .T * F][T \rightarrow .F][F \rightarrow \mathbf{id}.]$	$*$**id**
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow .T * F][T \rightarrow F.]$	$*$**id**
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow T. * F]$	$*$**id**
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow T * .F]$	**id**
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow T * .F][F \rightarrow .\mathbf{id}]$	**id**
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow T * .F][F \rightarrow \mathbf{id}.]$	
$[S \rightarrow .E][E \rightarrow E + .T][T \rightarrow T * F.]$	
$[S \rightarrow .E][E \rightarrow E + T.]$	
$[S \rightarrow E.]$	

Correctness

Lemma

If $([S' \to .S], uv) \vdash^*_{P_G} (\rho, v)$ *then* $hist(\rho) \overset{*}{\underset{G}{\Longrightarrow}} u$

Corollary: $L(P_G) \subseteq L(G)$

Lemma

Let $A \in V_N$ *and* $w \in V_T^*$. *If* $A \overset{*}{\underset{G}{\Longrightarrow}} w$, *there exists* $A \to \alpha \in P$ *such that for all* $\rho \in IT_G^*$ *and* $v \in V_T^*$

$$(\rho[A \to .\alpha], wv) \vdash^*_{P_G} (\rho[A \to \alpha.], v)$$

Corollary: $L(P_G) \supseteq L(G)$

Automaton with Output

A tuple $P = (V, Q, O, \Delta, q_0, F)$ where:

- input alphabet V, output alphabet O
- finite set of states Q, initial state $q_0 \in Q$, final states $F \subseteq Q$
- $\Delta \subseteq (Q^+ \times (V \cup \{\varepsilon\})) \times Q^* \times (O \cup \{\varepsilon\})$
- Alternatively:
  $\delta \colon (Q^+ \times (V \cup \{\varepsilon\})) \to 2^{Q^*} \times (O \cup \{\varepsilon\})$
  where δ is a partial function

- Essentially like a normal PDA but with output on steps

Left/Predictive/Top-Down Parser

$P'_G = (V_T, IT_G, P, \delta_I, [S' \to .S], \{[S' \to S.]\})$ where

$\delta_I([X \to \beta.Y\gamma], \varepsilon) = \{[X \to \beta.Y\gamma][Y \to .\alpha], Y \to \alpha) \mid Y \to \alpha \in P\}$

Configuration: $IT_G^+ \times V_T^* \times P^*$

Step:
$(\rho[X \to \beta.Y\gamma], w, o) \vdash_{P'_G} (\rho[X \to \beta.Y\gamma][Y \to .\alpha], w, o(Y \to \alpha))$

Right/Bottom-Up Parser

$P^r_G = (V_T, IT_G, P, \delta_r, [S' \to .S], \{[S' \to S.]\})$ where

$$\delta_r([X \to \beta.Y\gamma][Y \to \alpha.], \varepsilon) = \{[X \to \beta Y.\gamma], Y \to \alpha)\}$$

Configuration: $IT_G^+ \times V_T^* \times P^*$

Step:
$(\rho[X \to \beta.Y\gamma][Y \to \alpha.], w, o) \vdash_{P^r_G} (\rho[X \to \beta Y.\gamma], w, o(Y \to \alpha)))$

Deterministic Parsers

LL(k): Deterministic left parsers

- Read the input from left to right
- Find leftmost derivation
- Take decisions as early as possible, i.e. on expansion
- Use k symbols look ahead to decide about expansions

LR(k): Deterministic right parsers

- Read the input from left to right
- Find rightmost derivation in reverse order
- Delay decisions as long as possible, i.e. until reduction
- Use k tokens look ahead to
  - decide whether to shift or reduce (in "shift-reduce-conflicts")
  - decide by which rule to reduce (in "reduce-reduce-conflicts")

Example: Predictive Parser

$S' \to S, S \to aSb|\varepsilon$

- 1-symbol look ahead for expansions

TOS	**LA**	**new TOS**	**used production**
$([S' \to .S])$	\$	$\begin{pmatrix}[S \to .] \\ [S' \to .S]\end{pmatrix}$	$S \to \varepsilon$
$([S' \to .S])$	a	$\begin{pmatrix}[S \to .aSb] \\ [S' \to .S]\end{pmatrix}$	$S \to aSb$
$([S \to a.Sb])$	b	$\begin{pmatrix}[S \to .] \\ [S \to a.Sb]\end{pmatrix}$	$S \to \varepsilon$
$([S \to a.Sb])$	a	$\begin{pmatrix}[S \to .aSb] \\ [S \to a.Sb]\end{pmatrix}$	$S \to aSb$

- shift rules

TOS	**Input**	**new TOS**
$([S \to .aSb])$	a	$([S \to a.Sb])$
$([S \to aS.b])$	b	$([S \to aSb.])$

- reduction rules

TOS	**Input**	**new TOS**
$\begin{pmatrix} [S \to .] \\ [S' \to .S] \end{pmatrix}$	ε	$([S' \to S.])$
$\begin{pmatrix} [S \to aSb.] \\ [S' \to .S] \end{pmatrix}$	ε	$([S' \to S.])$
$\begin{pmatrix} [S \to .] \\ [S \to a.Sb] \end{pmatrix}$	ε	$([S \to aS.b])$
$\begin{pmatrix} [S \to aSb.] \\ [S \to a.Sb] \end{pmatrix}$	ε	$([S \to aS.b])$