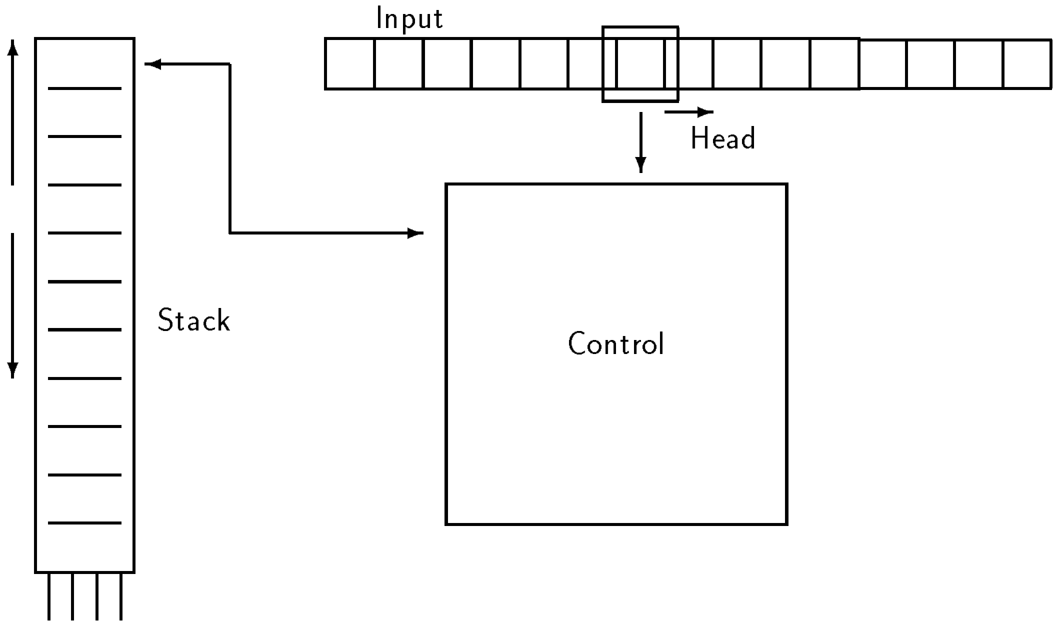
Pushdown Automata and Parser

Mooly Sagiv
Tel Aviv University
sagiv@math.tau.ac.il
and
Reinhard Wilhelm
Universität des Saarlandes
wilhelm@cs.uni-sb.de

May 13, 1997

Pushdown Automata



Example Automaton

Accepted language $L = \{a^i\, b^i \mid i \geq 0\}$
Context Free Grammar $S \rightarrow aSb|\varepsilon$
Pushdown automaton

top-stack	**input**		
	a	b	\$
(0)	$\begin{pmatrix}1\\0\end{pmatrix}$	(2)	(3)
(1)	$\begin{pmatrix}1\\1\end{pmatrix}$	ε	(2)

state 0: Initial state,

state 1: For next character

- **a:** Counts a's (pushing state 1),
- **b:** Counts b's (popping one state 1)

state 2: Error state,

state 3: Accepting (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:
  δ: $(Q^+ \times (V \cup \{\varepsilon\})) \rightarrow 2^{Q^*}$
  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 **step binary relation** on configurations is defined by:
  $(\gamma, aw) \vdash_P (\gamma', w)$ if
  - $\gamma \equiv \gamma_1\, \gamma_2$
  - $\gamma' \equiv \gamma_1\, \gamma_3$
  - $(\gamma_2, a, \gamma_3) \in \Delta$
- $\vdash_P^*$ 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 $a \in V_T$, $(\gamma_1, a, \gamma_2), (\gamma_1', a, \gamma_2') \in \Delta$ such that γ_1' is a suffix of γ_1 implies
  - $\gamma_1 = \gamma_1'$ and
  - $\gamma_2 = \gamma_2'$
- If $(\gamma_1, \varepsilon, \gamma_2), (\gamma_1', a, \gamma_2') \in \Delta$ such that γ_1' is a suffix of γ_1 or vice versa, then
  - $\gamma_1 = \gamma_1'$ and
  - $\gamma_2 = \gamma_2'$

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 $A \rightarrow \alpha\beta \in P$
- An item (A, α, β) is denoted by $[A \rightarrow \alpha.\beta]$
- Interpretation:

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

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

Extended Context Free Grammar

- New start symbol S'
- Additional production $S' \rightarrow S$

The Item Pushdown Automaton

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

top-stack	**inp.**	**new top-stack**	**comment**
$([X \to \beta.Y\gamma])$	ε	$\begin{pmatrix} [Y \to .\alpha] \\ [X \to \beta.Y\gamma] \end{pmatrix}$	$Y \to \alpha \in P$ "expand"
$([X \to \beta.a\gamma])$	a	$([X \to \beta a.\gamma])$	"shift"
$\begin{pmatrix} [Y \to \alpha.] \\ [X \to \beta.Y\gamma] \end{pmatrix}$	ε	$(X \to \beta Y.\gamma])$	"reduce"

- Sources of **nondeterminism**: expansion transitions; there may be several productions for Y.
- Different deterministic parsing methods remove nondeterminism in different ways.

Example:

1. $S' \to S$,

2. $S \to \varepsilon$,

3. $S \to aSb$

top-stack	input	new top-stack	comment
$([S' \to .S])$	ε	$\begin{pmatrix}[S \to .]\\ [S \to .S]\end{pmatrix}$	$e_{1,2}$
$([S' \to .S])$	ε	$\begin{pmatrix}[S \to .aSb]\\ [S \to .S]\end{pmatrix}$	$e_{1,3}$
$([S \to a.Sb])$	ε	$\begin{pmatrix}[S \to .]\\ [S \to a.Sb]\end{pmatrix}$	$e_{2,2}$
$([S \to a.Sb])$	ε	$\begin{pmatrix}[S \to .aSb]\\ [S \to a.Sb]\end{pmatrix}$	$e_{2,3}$
$([S \to .aSb])$	a	$([S \to a.Sb])$	s_1
$([S \to aS.b])$	b	$([S \to aSb.])$	s_2
$\begin{pmatrix}[S \to .]\\ [S' \to .S]\end{pmatrix}$	ε	$([S' \to S.])$	r_1
$\begin{pmatrix}[S \to aSb.]\\ [S' \to .S]\end{pmatrix}$	ε	$([S' \to S.])$	r_2
$\begin{pmatrix}[S \to .]\\ [S \to a.Sb]\end{pmatrix}$	ε	$([S \to aS.b])$	r_3
$\begin{pmatrix}[S \to aSb.]\\ [S \to a.Sb]\end{pmatrix}$	ε	$([S \to aS.b])$	r_4

Top-Stack	Input	New Top-Stack
$[S \to .E]$	ε	$[S \to .E][E \to .E+T]$
$[S \to .E]$	ε	$[S \to .E][E \to .T]$
$[E \to .E+T]$	ε	$[E \to .E+T][E \to .E+T]$
$[E \to .E+T]$	ε	$[E \to .E+T][E \to .T]$
$[F \to (.E)]$	ε	$[F \to (.E)][E \to .E+T]$
$[F \to (.E)]$	ε	$[F \to (.E)][E \to .T]$
$[E \to .T]$	ε	$[E \to .T][T \to .T*F]$
$[E \to .T]$	ε	$[E \to .T][T \to .F]$
$[T \to .T*F]$	ε	$[T \to .T*F][T \to .T*F]$
$[T \to .T*F]$	ε	$[T \to .T*F][T \to .F]$
$[E \to E+.T]$	ε	$[E \to E+.T][T \to .T*F]$
$[E \to E+.T]$	ε	$[E \to E+.T][T \to .F]$
$[T \to .F]$	ε	$[T \to .F][F \to .(E)]$
$[T \to .F]$	ε	$[T \to .F][F \to .\mathbf{id}]$
$[T \to T*.F]$	ε	$[T \to T*.F][F \to .(E)]$
$[T \to T*.F]$	ε	$[T \to T*.F][F \to .\mathbf{id}]$
$[F \to .(E)]$	(	$[F \to (.E)]$
$[F \to .\mathbf{id}]$	**id**	$[F \to \mathbf{id}.]$
$[F \to (E.)]$	)	$[E \to (E).]$
$[E \to E.+T]$	$+$	$[E \to E+.T]$
$[T \to T.*F]$	$*$	$[T \to T*.F]$
$[T \to .F][F \to \mathbf{id}.]$	ε	$[T \to F.]$
$[T \to T*.F][F \to \mathbf{id}.]$	ε	$[T \to T*F.]$
$[T \to .F][F \to (E).]$	ε	$[T \to F.]$
$[T \to T*.F][F \to (E).]$	ε	$[T \to T*F.]$
$[T \to .T*F][T \to F.]$	ε	$[T \to T.*F]$
$[E \to .T][T \to F.]$	ε	$[E \to T.]$
$[E \to E+.T][T \to F.]$	ε	$[E \to E+T.]$
$[E \to E+.T][T \to T*F.]$	ε	$[E \to E+T.]$
$[T \to .T*F][T \to T*F.]$	ε	$[T \to T.*F]$
$[E \to .T][T \to T*F.]$	ε	$[E \to T.]$
$[F \to (.E)][E \to T.]$	ε	$[F \to (E.)]$
$[F \to (.E)][E \to E+T.]$	ε	$[F \to (E.)]$
$[E \to .E+T][E \to T.]$	ε	$[E \to E.+T]$
$[E \to .E+T][E \to E+T.]$	ε	$[E \to E.+T]$
$[S \to .E][E \to T.]$	ε	$[S \to E.]$
$[S \to .E][E \to E+T.]$	ε	$[S \to E.]$

Accepting $id + id * id$

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

The Simulation Lemma

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

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

The Other Direction

Lemma. *Let* $A \in V_N$ *and* $w \in V_T^*$.
For every $m \geq 1$, *if* $A \overset{m}{\underset{G}{\Longrightarrow}} w$, *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 $P = (V, Q, \Delta, O, q_0, F)$ where:

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

Left/Predictive/Top-Down Parser

$$P_G^l = (V_T, IT_G, P, \delta_l, [S' \to .S], \{[S' \to S.]\})$$ where

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

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

$$(\rho[X \to \beta.Y\gamma], w, o) \vdash_{P_G^l}$$
$$(\rho([X \to \beta.Y\gamma][Y \to .\alpha], w, o(Y \to \alpha))$$

Right/Bottom-Up Parser

$$P_G^r = (V_T, IT_G, P, \delta_r, [S' \rightarrow .S], \{[S' \rightarrow S.]\})$$ where

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

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

$$(\rho[X \rightarrow \beta.Y\gamma][Y \rightarrow \alpha.], w, o) \vdash_{P_G^r}$$
$$(\rho([X \rightarrow \beta Y.\gamma], w, o(Y \rightarrow \alpha)))$$

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

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

- 1-symbol look ahead for expansions

top-stack	LA	new top-stack	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

top-stack	Input	new top-stack
$([S \to .aSb])$	a	$([S \to a.Sb])$
$([S \to aS.b])$	b	$([S \to aSb.])$

- reduction rules

top-stack	Input	new top-stack
$\begin{pmatrix}[S \rightarrow .] \\ [S' \rightarrow .S]\end{pmatrix}$	ε	$([S' \rightarrow S.])$
$\begin{pmatrix}[S \rightarrow aSb.] \\ [S' \rightarrow .S]\end{pmatrix}$	ε	$([S' \rightarrow S.])$
$\begin{pmatrix}[S \rightarrow .] \\ [S \rightarrow a.Sb]\end{pmatrix}$	ε	$([S \rightarrow aS.b])$
$\begin{pmatrix}[S \rightarrow aSb.] \\ [S \rightarrow a.Sb]\end{pmatrix}$	ε	$([S \rightarrow aS.b])$