Parallel Communicating Automata Systems

by

M. Sakthi Balan

Data Processing Lab
Department of Computer Science
University of Western Ontario
Contents of the Talk

- Grammar Systems

Conclusion
Contents of the Talk

Contents

❏ Grammar Systems

❏ Parallel Communicating Grammar Systems
Contents of the Talk

Contents

- Grammar Systems
- Parallel Communicating Grammar Systems
- Parallel Communicating Automata Systems
Contents of the Talk

Contents

- Grammar Systems
- Parallel Communicating Grammar Systems
- Parallel Communicating Automata Systems
- Parallel Communicating Finite State Automata
Contents of the Talk

- Grammar Systems
- Parallel Communicating Grammar Systems
- Parallel Communicating Automata Systems
- Parallel Communicating Finite State Automata
- Parallel Communicating Pushdown Automata

DPL Lab, Dept. CS, UWO
March 24, 2005
M. Sakthi Balan
Contents

- Grammar Systems
- Parallel Communicating Grammar Systems
- Parallel Communicating Automata Systems
- Parallel Communicating Finite State Automata
- Parallel Communicating Pushdown Automata
- Conclusion
Grammar Systems

- System consisting of finite number of grammars working together under a specified protocol to generate languages.

- Cooperating Distributed (CD) Grammar Systems.
  - Black-board Model.
  - At any time only one component is active.
  - Modes of derivations: $t$-mode, $*$-mode, $= k$-mode, $\leq k$ mode and $\geq k$-mode.

- Parallel Communicating (PC) Grammar Systems.
  - Classroom Model.
  - All components work in parallel.
  - Different variants: $\{\text{centralized, non-centralized}\} \times \{\text{returning, non-returning}\}$.
Many FSA working together.

Communication by states.

The querying component imparts a query symbol in its state information specific to the component to be queried and gets the state of the queried component after communication gets over.
- Finite number of PDA working together.
- Communication by stack
- The querying component imparts a query symbol in its stack specific to the component to be queried and gets the stack information of the queried component in its stack after communication gets over.
- There should not be any cyclic querying.
- Centralized: Only one component can query
- Non-centralized: Any component can query
- Returning: Once the communication takes place the queried component loses all the stack information and so again starts from its start stack symbol.
- Non-Returning: The stack of the queried component retains the copy even after the communication.

So totally we have four variants

\[ \{ \text{centralized, non-centralized} \} \times \{ \text{returning, non-returning} \}. \]
Cooperating Distributed (CD) \(\{FSA, PDA\}\) Systems

- \(CDFSA\) - no increase in power
- \(CDPDA\) - equivalent to Turing machine in all the modes of acceptance

Parallel Communicating (PC) \(\{FSA, PDA\}\) Systems

- \(PCFSA\) - equivalent to multi-head finite state automata
- \(PCPDA\) - equivalent to Turing machine in all variants, except one which is still open
A parallel communicating pushdown automata system of degree $n$ is a construct

$$
\mathcal{A} = (V, \Delta, A_1, A_2, \ldots, A_n, K), n \geq 1
$$

where $V$ is the input alphabet, $\Delta$ is the alphabet of pushdown symbols, for each $1 \leq i \leq n$, $A_i = (Q_i, V, \Delta, f_i, q_i, Z_i, F_i)$ is a pushdown automaton with the set of states $Q_i$, the initial state $q_i \in Q_i$, the alphabet of input symbols $V$, the alphabet of pushdown symbols $\Delta$, the initial pushdown symbols $Z_i \in \Delta$, the set of final states $F_i \subseteq Q_i$, and the transition mapping $f_i$ from $Q_i \times (V \cup \{\epsilon\}) \times \Delta$ into the finite subsets of $Q_i \times \Delta^*$, $K \subseteq \{K_1, K_2, \ldots, K_n\} \subseteq \Delta$ is the set of query symbols.

The automata $A_1, A_2, \ldots, A_n$ are called the components of the system $\mathcal{A}$. 
If there exists only one $i, 1 \leq i \leq n$, such that for $A_i, (r, \alpha) \in f_i(q, a, A)$ with $\alpha \in \Delta^*, |\alpha|_K > 0$ for some $r, q \in Q_i, a \in V \cup \{\epsilon\}, A \in \Delta$, then the system is said to be *centralized* and $A_i$ is said to be the *master* of the system, i.e only one of the component, called the the master, is allowed to introduce queries. For the sake of simplicity, whenever a system is centralized its master is taken to be the first component.

We define a configuration of a parallel communicating pushdown automata system as a $3n$-tuple

$$(s_1, x_1, \alpha_1, s_2, x_2, \alpha_2, \ldots, s_n, x_n, \alpha_n)$$

where for $1 \leq i \leq n$, $s_i \in Q_i$ is the current state of the component $A_i, x_i \in V^*$ is the remaining part of the input word which has not yet been read by $A_i, \alpha_i \in \Delta^*$ is the contents of the $i$th stack, its first letter being the topmost symbol.
The initial configuration of a parallel communicating pushdown automata system is defined as

\[(q_1, x, Z_1, q_2, x, Z_2, \cdots, \cdots, q_n, x, Z_n)\]

where \(q_i\) is the initial state of the component \(i\), \(x\) is the input word, and \(Z_i\) is the initial stack symbol of the component \(i\), \(1 \leq i \leq n\). It should be noted here that all the components receive the same input word \(x\).
Two variants of transition relations:

First one: \( (s_1, x_1, B_1 \alpha_1, \ldots, s_n, x_n, B_n \alpha_n) \vdash (p_1, y_1, \beta_1, \ldots, p_n, y_n, \beta_n) \)

where \( B_i \in \Delta, \alpha_i, \beta_i \in \Delta^*, 1 \leq i \leq n \), iff one of the following two conditions holds:

(i) \( K \cap \{B_1, B_2, \ldots, B_n\} = \emptyset \) and \( x_i = a_i y_i, a_i \in V \cup \{\epsilon\}, (p_i, \beta'_i) \in f_i(s_i, a_i, B_i), \beta_i = \beta'_i \alpha_i, 1 \leq i \leq n \),

(ii) (a) for all \( i, 1 \leq i \leq n \) such that \( B_i = K_{j_i} \) and \( B_{j_i} \notin K, \beta_i = B_{j_i} \alpha_{j_i} \alpha_i \),

(b) for all other \( r, 1 \leq r \leq n \), \( \beta_r = B_r \alpha_r \), and

(c) \( y_t = x_t, p_t = s_t \), for all \( t, 1 \leq t \leq n \).
Second one:

\[(s_1, x_1, B_1 \alpha_1, \ldots, s_n, x_n, B_n \alpha_n) \vdash_r (p_1, y_1, \beta_1, \ldots, p_n, y_n, \beta_n),\]

where \(B_i \in \Delta, \alpha_i, \beta_i \in \Delta^*, 1 \leq i \leq n\), iff one of the following two conditions holds:

(i) \(K \cap \{B_1, B_2, \ldots, B_n\} = \emptyset\) and \(x_i = a_i y_i, a_i \in V \cup \{\epsilon\}, (p_i, \beta'_i) \in f_i(s_i, a_i, B_i), \beta_i = \beta'_i \alpha_i, 1 \leq i \leq n\),

(ii) (a) for all \(1 \leq i \leq n\) such that \(B_i = K_{ji}\) and \(B_{ji} \notin K\), \(\beta_i = B_{ji} \alpha_ji \alpha_i\), and \(\beta_{ji} = Z_{ji}\),

(b) for all the other \(r, 1 \leq r \leq n, \beta_r = B_r \alpha_r\), and

(c) \(y_t = x_t, p_t = s_t\), for all \(t, 1 \leq t \leq n\).

The communication between the components has more priority than the usual transitions in individual components. So, whenever a component has a query symbol in the top of its stack it has to be satisfied by the requested component.
The stack contents of the sender is retained in the case of relation $\leftarrow$, whereas it loses all the symbols in the case of $\leftarrow_r$.

A parallel communicating pushdown automata system whose computations are based on relation $\leftarrow$ is said to be non-returning; if its computations are based on relation $\leftarrow_r$ it is said to be returning.

There are four variants of PCPDA
The language accepted by a parallel communicating pushdown automata system, \( \mathcal{A} \) is defined as

\[
L(\mathcal{A}) = \{ x \in V^* \mid (q_1, x, Z_1, \ldots, q_n, x, Z_n) \vdash^* (s_1, \epsilon, \alpha_1, \ldots, s_n, \epsilon, \alpha_n), \\
\quad s_i \in F_i, 1 \leq i \leq n \},
\]

\[
L_r(\mathcal{A}) = \{ x \in V^* \mid (q_1, x, Z_1, \ldots, q_n, x, Z_n) \vdash^* (s_1, \epsilon, \alpha_1, \ldots, s_n, \epsilon, \alpha_n), \\
\quad s_i \in F_i, 1 \leq i \leq n \}
\]

where \( \vdash^* \) and \( \vdash_r^* \) denote the reflexive and transitive closure of \( \vdash \) and \( \vdash_r \) respectively.
Formal Definition of PCPDA

- \( \text{rcpcpa}(n) \) - for returning centralized parallel communicating pushdown automata systems of degree at most \( n \),

- \( \text{rpcpa}(n) \) - for returning non-centralized parallel communicating pushdown automata systems of degree at most \( n \),

- \( \text{cpcpa}(n) \) - for centralized parallel communicating pushdown automata systems of degree at most \( n \),

- \( \text{pcpa}(n) \) - for parallel communicating pushdown automata systems of degree at most \( n \).

If \( x(n) \) is a type of automata system, then \( X(n) \) is the class of languages accepted by pushdown automata systems of type \( x(n) \).
\begin{itemize}
  \item $PCPA(2)$ and $RPCPA(3)$ equals $RE$.
  
  \item $CPCPA(3)$ equals $RE$.
  
  \item $RCPCPA(k)$ is at least the power of $k$-head pushdown automata.
  
  \item $RCPCPA(2)$ accepts non-$ETOL$ languages, and hence $RPCPA(2)$ accepts non-$ETOL$ languages.
  
  \item What is the exact power of $RCPCPA/RPCPA(2)$ is it equivalent to $RE$ or properly contained in $RE$?
\end{itemize}
☐ Parallel communicating pushdown automata systems with filters

☐ Parallel communicating pushdown automata systems with $k$-symbol communication.

☐ Defining automata systems (FSA, PDA) with only synchronizing symbols and with no communication. What will be the power of this model?