Distributed Processing in Automata

by

M. Sakthi Balan

Theoretical Computer Science Lab

Department of Computer Science and Engineering

Indian Institute of Technology, Madras

2000

Contents

Introduction: Grammar Systems

- The grammars and automata are the classic computing devices in Formal Language and Automata theory. Here the computation is accomplished by only one agent.
- Theory of grammar system is a very abstract model for the distributed computation which plays a major role in modern computer science.
- A grammar system is a set of grammars working in unison, according to a specified protocol, to generate one language. The main reason is to model distribution, to increase the generative power, to decrease the complexity, etc.

  Two types: sequential(CD grammar systems) or parallel(PC Grammar Systems).

- A comprehensive treatment of grammar systems and a survey of the recent developments in this area can be found in *Grammar Systems. A Grammatical Approach to Distribution and Cooperation,*authored by E. Csuhaj-Varju, J.Dassow, J.Keleman and Gh. Paun.

- The study of sequential grammar systems(also called *Cooperating Distributed(CD)Grammar Systems*) was initiated by *Csuhaj-Varju* and *Dassow*. The explicit notion of CD grammar systems was introduced by *Meersman et.al.*.

- *Csuhaj-Varju et.al.*introduced *Parallel Communicating(PC) Grammar Systems*).

- Modes of derivations: t-mode, *-mode, $=$ k-mode, $\leq$ k-mode and $\geq$ k-mode

Introduction: DNA Computing

- As we have Chomskian Grammars for electronic computers we have sticker systems for the future computers: DNA Computers.
- Machine equivalent of the chomskian grammars is the automata, for the sticker systems it is the Watson-Crick automata.
- Classic automata work on a single tape. Watson-Crick automata work on two tapes(double stranded)in a correlated manner through complementary property.
- Powers of DNA Computing: Watson-Crick Complementarity and massive parallelism.
- Watson-Crick automata uses only the first power. Modeling the second feature is still an open area.

Motivation

- Even though the work done in the area of distributed rewriting system is enormous, not much work has been done in the area of distributed analytical machine model. So, we proceed to:
  - Develop a distributed finite state automata and distributed pushdown automata in all the known modes of acceptance, study their accepting power, and thereby studying the hierarchy arising from it.
  - Develop a distributed Watson-Crick finite state automata and distributed Watson-Crick pushdown automata and study its accepting power in all its variants in all the modes of acceptance.

What we have to do..

- Define distributed finite state automata (deterministic and nondeterministic), distributed pushdown automata (deterministic and nondeterministic), Watson-Crick finite state automata in the distributed sense and define the Watson-Crick pushdown automata (both stand alone and distributed) and study its power.
- Study the accepting power of the proposed model with respect to the different modes of acceptance and there by study its hierarchy.
- Study the accepting power of the proposed model and compare it with the chomskian hierarchy.

n-FSA Definition

- An n-FSA is a 5-tuple $\Gamma = (Q, V, \Delta, q_0, F)$ where, $Q = n$-tuple $(Q_1, Q_2, \cdots, Q_n)$ where each Q_i is a set of states of the ith component, V=finite set of alphabet, $\Delta =$ n-tuple $(\delta_1, \delta_2, \cdots, \delta_n)$ of state transition functions where each $\delta_i : Q_i \times (V \cup \{\lambda\}) \longrightarrow 2^{Q_{union}}$, $1 \leq i \leq n$, $q_0 \in Q_{union}$ is the initial state, $F \subseteq Q_{union}$ is the set of final accepting states, where $Q_{union} = \cup_i Q_i$.
- Mode of acceptance
  1. t-mode: change only after termination.
  2. *-mode: change at any time if possible.
  3. $= k$-mode: change after exactly k steps.
  4. $\leq k$-mode: change after $k'(k' \leq k)$ steps.
  5. $\geq k$-mode: change after $k'(k' \geq k)$ steps.

Modes of Acceptance: Explanation

- **t-mode** In component i the system follows its transition function as any "stand alone" FSA. The control is tranfered from the component i to component j only if the system arrives at a state $q \notin Q_i$ and $q \in Q_j$. The selection of j is nondeterministic if q belongs to more than one Q_j.
- **Instantaneous description of the n-FSA working in t-mode** (ID) is given by a 3-tuple (q, w, i) where $q \in Q,\ w \in V^*$. Here, q denotes the current state of the whole system, w the portion of the input string yet to be read and i the index of the component in which the system is currently in.

Contd..

- The transition between the ID's is defined as follows:

  1. $(q, aw, i) \vdash (q', w, i)$ iff $q' \in \delta_i(q, a)$ where $q, q' \in Q_i, a \in V \cup \{\lambda\}, w \in V^*, 1 \leq i \leq n$
  2. $(q, w, i) \vdash (q, w, j)$ iff $q \in Q_j - Q_i$

- Let $\vdash^*$ be the reflexive and transitive closure of $\vdash$.

- The language accepted by the n-FSA $\Gamma = (Q, V, \Delta, q_0, F)$ working in t-mode is defined as follows,

$$L_t(\Gamma) = \{w \in V^* \mid (q_0, w, i) \vdash (q_f, \lambda, j)$$

$$\text{for some } q_f \in F \ 1 \leq j, i \leq n \text{ and } q_0 \in Q_i\}$$

Contd..

- Instantaneous description(ID) of the n-FSA working in $= k$-mode is given by a 4-tuple (q, w, i, j) where $q \in Q_{union},\ w \in V^*$, $1 \leq i \leq n$, j is a non negative integer.
- In this ID of the n-FSA, j denotes the number of steps for which the system has been in, in the ith component.
- We accept the strings only if the n-FSA is in the final state in some component i after reading the string and provided ; it has completed k steps in the component i in the case of $= k$-mode of acceptance.
- We denote by $L_\alpha(M)$ the language accepted by the n-FSA M working in $\alpha \in \{t, *\} \cup \{= k, \leq k, \geq k \mid k \geq 1\}$-mode.

Results in n-FSA

- For any n-FSA Γ, $n \geq 1$ working in α-mode, we have $L_\alpha \in REG$.
- **Proof for t-mode** Let $\Gamma = (Q, V, \Delta, q_0, F)$ be a n-FSA working in t-mode where $\Delta = (\delta_1, \delta_2, \cdots, \delta_n)$ and $Q = Q_1, Q_2, \cdots, Q_n$. Construct the FSA $M = (Q', V, \delta, q_0', F')$ as follows:
  - $Q' = \{[q, i] \mid q \in Q_{union},\ 1 \leq i \leq n\} \cup \{q_0'\}$
  - $F' = \{[q_f, i] \mid q_f \in F,\ 1 \leq i \leq n\}$
  - $[q_0, i'] \in \delta(q_0', \lambda)$ such that $q_0 \in Q_{i'}$
  - if $q_k \in Q_i$ then $[q_k, i] \in \delta([q_j, i], a)$
  - if $q_k \in Q_j - Q_i$ then $[q_k, j] \in \delta([q_j, i], \lambda),\ 1 \leq j \leq n$

Contd..

- **Proof for $= k$-mode** Construct the FSA $M = (Q', V, \delta, q_0', F')$ where,
  - $Q' = \{[q, i, j] \mid q \in Q_{union},\ 1 \leq i \leq n,\ 0 \leq j \leq k\}$
  - $F' = \{[q_f, i, k] \mid q_f \in F,\ 1 \leq i \leq n\}$
  - $[q_0, i', 0] \in \delta(q_0', \lambda)$ such that $q_0 \in Q_{i'}$
  - if $j < k$ then $[q_y, i, j] \in \delta([q_s, i, j-1], a)$
  - if $j = k$ then $[q_s, j', 0] \in \delta([q_s, i, k], \lambda), 1 \leq j' \leq n$ and $q_s \in Q_{j'}$.
- **Note** The results are in line with the results for CD Grammar Systems with Regular rules.

n-PDA Definition

- An n-PDA is a 7-tuple $M = (Q, V, \Gamma, \Delta, q_0, Z, F)$ where,

  1. Q, V, q_0, F are as in the definition of n-FSA.
  2. Γ is an n-tuple $(\Gamma_1, \Gamma_2, \cdots, \Gamma_n)$ where each Γ_i's are finite set of stack symbols.
  3. Δ is an n-tuple $(\delta_1, \delta_2, \cdots, \delta_n)$ of state transition functions where each $\delta_i : Q_i \times (V \cup \{\lambda\}) \times \Gamma_i \longrightarrow 2^{Q_{union} \times \Gamma_i^*}$, $1 \leq i \leq n$
  4. Z is an n-tuple $(Z_1, Z_2, \cdots, Z_n)$ where each $Z_i \in \Gamma_i$ $(1 \leq i \leq n)$ is the start symbol of stack for the ith component.

Contd..

- **Nondeterministic Automata:** Every component is nondeterministic locally and the selection of component is also nondeterministic.
- **Deterministic Automata:** Every component is deterministic locally but the selection of component is nondeterministic.
- **Completely Deterministic:**Both are deterministic.
- **Note:** Determinism or Nondeterminism does not matter in the case of the distributed finite state automata since deterministic finite state automata and nondeterministic finite state automata are equivalent.

Contd..

- **Instantaneous description(ID)** of the n-PDA is given by a n+3-tuple $(q, w, \alpha_1, \alpha_2, \cdots, \alpha_n, i)$ where q, w, i are the same as earlier ones and $\alpha_1, \alpha_2, \cdots, \alpha_n$ are the contents of the stacks of the components respectively.
- The transition between the ID's is defined as follows:
  - $(q, aw, \alpha_1, \alpha_2, \cdots, X\alpha_i, \cdots, \alpha_n, i) \vdash$
    $(q', w, \alpha_1, \alpha_2, \cdots, \beta\alpha_i, \cdots, \alpha_n, i)$
    iff $(q', \beta) \in \delta_i(q, a, X)$ where $q, q' \in Q_i$.
  - $(q, w, \alpha_1, \alpha_2, \cdots, \alpha_n, i) \vdash (q, w, \alpha_1, \alpha_2, \cdots, \alpha_n, j)$
    iff $q \in Q_j - Q_i,\ w \in V^*,\ 1 \leq i, j \leq n,\ \alpha_i \in \Gamma_i^*$.

Contd..

- The language accepted by the n-PDA M in the t-mode is $L_t(M) = \{w \in V^* \mid (q_o, w, Z_1, Z_2, \cdots, Z_n, i') \vdash^* (q_f, \lambda, \alpha_1, \alpha_2, \cdots, \alpha_n, i)$ for some $q_f \in F,\ 1 \leq i, i' \leq n,\ \alpha_i \in \Gamma_i^*$ and $q_0 \in Q_{i'}\}$ Above language is called the language accepted by final state. Like wise we can define the language accepted by empty store.

- ID for the *-mode is similar. For the other modes the ID will include another component which counts the number of transitions the system has gone through in a specific component.

- We denote by $L_\alpha(M)$ the language accepted by the n-PDA M working in $\alpha \in \{t, *\} \cup \{= k, \leq k, \geq k \mid k \geq 1\}$-mode.

Example

- Consider the 2-PDA $M = ((Q_1, Q_2), V, (\Gamma_1, \Gamma_2), (\delta_1, \delta_2), \{q_0\}, (Z_1, Z_2), \{q_f\})$ where
- $Q_1 = \{q_0, q_a, q_b, q_T\}, Q_2 = \{q_{a'}, q_{b'}, q_s, q_f\}, V = \{a, b\}, F = \{q_f\}$
- $\Gamma_1 = \Gamma_2 = \{Z, a, b\}, \Delta = (\delta_1, \delta_2)$
- $\delta_1(q_0, a, Z_1) = \{(q_a, aZ_1)\}$
- $\delta_1(q_0, b, Z_1) = \{(q_b, bZ_1)\}$
- $\delta_1(q_a, b, X) = \{(q_a, bX)\}$
- $\delta_1(q_b, a, X) = \{(q_b, aX)\}$
- $\delta_1(q_a, a, X) = \{(q_T, X), (q_a, aX)\}$
- $\delta_1(q_b, b, X) = \{(q_T, X), (q_b, bX)\}$
- $\delta_1(q_T, \lambda, a) = \{(q_{a'}, \lambda)\}$
- $\delta_1(q_T, \lambda, b) = \{(q_{b'}, \lambda)\}$

- $\delta_2(q_{a'}, \lambda, Y) = \{(q_T, aY)\}$
- $\delta_2(q_{b'}, \lambda, Y) = \{(q_T, bY)\}$
- $\delta_1(q_T, \lambda, Z_1) = \{(q_e, Z_1)\}$
- $\delta_2(q_e, \lambda, \{a, b\}) = \{(q_s, \lambda)\}$
- $\delta_2(q_s, a, a) = \{(q_s, \lambda)\}$
- $\delta_2(q_s, b, b) = \{(q_s, \lambda)\}$
- $\delta_2(q_s, \lambda, Z_2) = \{(q_f, Z_2)\}$
- $\delta_2(q_f, \lambda, Z_2) = \{(q_f, Z_2)\}$

$$L_{t,*} \;=\; \{ww \mid w \in \{a, b\}^*\}.$$

Results

- **Deterministic and Nondeterministic n-PDA:** every mode is equivalent. Acceptance by final states is equivalent to acceptance by empty store in all the modes.
- **Nondeterministic n-PDA:** in all modes of acceptance, even two components are enough to get the power of a Turing machine.
- **Deterministic n-PDA:** in t-mode even two components make the system as powerful as a Turing machine. In other modes of acceptance, the system requires four components to get the power of Turing machine.
- **Completely Deterministic Distributed PDA:** equivalent to deterministic PDA.

Proofs: Distributed Nondeterministic PDA

- ***-mode $\Longrightarrow$ t-mode**
  - $Q'_i = Q_i \cup \{q_{ji} \mid \forall q \in Q_i \cap Q_j,\ 1 \leq j \leq n,\ j \neq i\},\ 1 \leq i \leq n$
  - $\delta'_i(q, a, X)$ includes all elements of $\delta_i(q, a, X)$ for all $q \in Q_i, a \in V, X \in \Gamma_i$.
  - If $q \in Q_i \cap Q_j$ $1 \leq i, j \leq n,\ i \neq j$ then
    1. $(q_{ij}, Y) \in \delta'_i(p, a, X)$ if $(q, Y) \in \delta_i(p, a, X)$, $a \in V$ and $X \in \Gamma_i^*$
    2. $(q, X) \in \delta'_j(q_{ij}, \lambda, X)$ $X \in \Gamma_j^*$

Contd..

- **t-mode $\Longrightarrow$ *-mode**

  1. if $q_0 \in Q_i$ then $Q'_i = \{[q,i] \mid 1 \leq i \leq n, q \in Q_i\} \cup \{q^* \mid q \in Q_i\} \cup \{q_0\}$.
  2. if $q_0 \notin Q_i$ then $Q'_i = \{[q,i] \mid 1 \leq i \leq n, q \in Q_i\} \cup \{q^* \mid q \in Q_i\}$.
  3. $\delta'_i(q_0, \lambda, X) = ([q_0, i], X)$
  4. $\delta'_i([q,i], a, X) = ([p,i], \gamma X)$ if $\delta_i(q, a, X) = (p, \gamma X)$ and $p \in Q_i$.
  5. $\delta'_i([q,i], a, X) = (p^*, \gamma X)$ if $\delta_i(q, a, X) = (p, \gamma X)$ and $p \notin Q_i$.
  6. $\delta'_j(p^*, \lambda, X) = ([p,j], X)$
  7. $F' = \{[q,i] \mid q \in F, 1 \leq i \leq n\}$

  Therefore, t-mode and *-mode are equivalent.

Contd..

- $= k$**-mode** $\Longrightarrow t$**-mode**

  1. $Q'_i = \{[q,x] \mid q \in Q_i \text{ and } 1 \leq x \leq k\} \cup \{q_{ji} \mid q \in Q_i \cap Q_j,\ 1 \leq j \leq n \text{ and } i \neq j\}$
  2. $([s, l+1], y) \in \delta'_i([p,l], a, X)$ provided $(s,y) \in \delta'_i(p,a,X)$ and $l < k$
  3. If $l = k$ then $(s_{ij}, y) \in \delta'_i([q,l], a, X)$ if $s \in Q_i \cap Q_j$ for $X \in \Gamma_i$ and $(s,y) \in \delta_i(q,a,X)$
  4. $([q,1], X) \in \delta'_j(q_{ij}, \lambda, X)$ for $X \in \Gamma_j$
  5. $F' = \{[q,k] \mid q \in F\}$

Contd..

- **t-mode $\Longrightarrow=$ k-mode**

  1. $Q'_i = Q_i \cup \{[q,s] \mid 1 \leq s \leq k,\ q \in Q_i\} \cup \{r'_{qi} \mid q \in Q_i\} \cup \{r_{q_{t,s}} \mid q \in Q_{T_i},\ 1 \leq i \leq n,\ q_t \notin Q_i$ is the state which can be arrived from the state $q, 1 \leq s \leq k,\ 1 \leq t \leq t_1$, where t_1 is a positive integer$\} \cup \{p_{i_1}, p_{i_2}, \cdots, p_{i_k}\}$ for $1 \leq i \leq n$.
  2. $Q'_{n+1} = \{r_{qi} \mid q \in Q_i, 1 \leq i \leq n\} \cup \{d_2, d_3, \cdots, d_k\}$.
  3. $Q'_{n+2} = \{f_1, f_2, \cdots, f_k\}$.
  4. $\Gamma'_i = \Gamma_i$ for $1 \leq i \leq n$.
  5. $\Gamma'_{n+1} = \{Z_{qi} \mid q \in Q_i, 1 \leq i \leq n\} \cup \{Z_{n+1}\}$.
  6. $\Gamma'_{n+2} = \{Z_{n+2}\}$.

Let Q_{T_i} be the set of all states q in $Q - Q_i$ such that q is a transition state for the component i and let $N(Q_{T_i})$ be the set of all states q in Q_i such that from q the system can arrive at a state $p \in Q_{T_i}$ by a single transition. We define the transition function $\delta'_i (1 \leq i \leq n)$ as follows,

1. $([q,1], YX) \in \delta'_i(q_0, a, X)$ if $(q, YX) \in \delta_i(q_0, a, X)$ $\forall i$ s.t $q_0 \in Q_i,\ 1 \leq i \leq n$.
2. $([q,1], X) \in \delta'_i(q, \lambda, X)\ \forall i,\ 1 \leq i \leq n$.
3. $([q,1], X) \in \delta'_i(r'_{qi}, \lambda, Y)$.
4. $(p_{i_{k-s+2}}, Z_{qi}X) \in \delta'_i(r_{q_{t,s}}, a, X)$.
5. $(p_{i_{j-1}}, Z_{qi}) \in \delta'_i(p_{i_j}, \lambda, Z_{qi})$ for $2 \leq j \leq k$.
6. $(q, \lambda) \in \delta_i(p_{i_1}, \lambda, Z_{qi})$.

Case 1 Let $q \notin F$

1. if $q \in N(Q_{T_i})$ and $s < k-1$ then $\{(r_{q_{t,s}}, X) \mid$ t is a finite integer $\geq 1\} \cup \{([p, s+1], YX)\} \in$

$\delta_i'([q,s],a,X)$ if $(p,YX) \in \delta_i(q,a,X)$.

2. if $q \in N(Q_{T_i})$ and $s = k-1$ then $(p,YX) \in \delta_i'([q,s],a,X)$ if $(p,YX) \in \delta_i(q,a,X)$.

3. if $q \notin N(Q_{T_i})$ and $s < k-1$ then $([p,s+1],YX) \in \delta_i'([q,s],a,X)$ if $(p,YX) \in \delta_i(q,a,X)$.

4. if $q \notin N(Q_{T_i})$ and $s = k-1$ then $(r_{qi},X) \in \delta_i([q,s],a,X)$.

Case 2 Let $q \in F$

1. if $q \in N(Q_{T_i})$ and $s < k-1$ then $(f_{k-s+1},X) \in \delta_i'([q,s],a,X)$.

2. if $q \in N(Q_{T_i})$ and $s = k-1$ then $(p,YX) \in \delta_i'([q,s],a,X)$ if $(p,YX) \in \delta_i'((q,a,X)$.

3. if $q \notin N(Q_{T_i})$ and $s < k-1$ then $\{(f_{k-s+1},X), ([p,s+1],YX)\} \in \delta_i'([q,s],a,X)$ if $(p,YX) \in \delta_i((q,a,X)$.

4. if $q \notin N(Q_{T_i})$ and $s = k-1$ then $\{(f_1,X),(r_{qi},X)\}$

$\in \delta'_i([q,s],a,X)$.

δ'_{n+1} definition is as follows,

1. $(d_2, Z_{qi}Z_{n+1}) \in \delta'_{n+1}(r_{qi}, \lambda, Z_{n+1})$.
2. $(d_{i+1}, Y) \in \delta'_{n+1}(d_i, \lambda, Y)$ for $2 \leq i \leq k-1$.
3. $(r'_{qi}, Z_{n+1}) \in \delta'_{n+1}(d_k, \lambda, Z_{qi}Z_{n+1})$.

δ'_{n+2} is defined as,

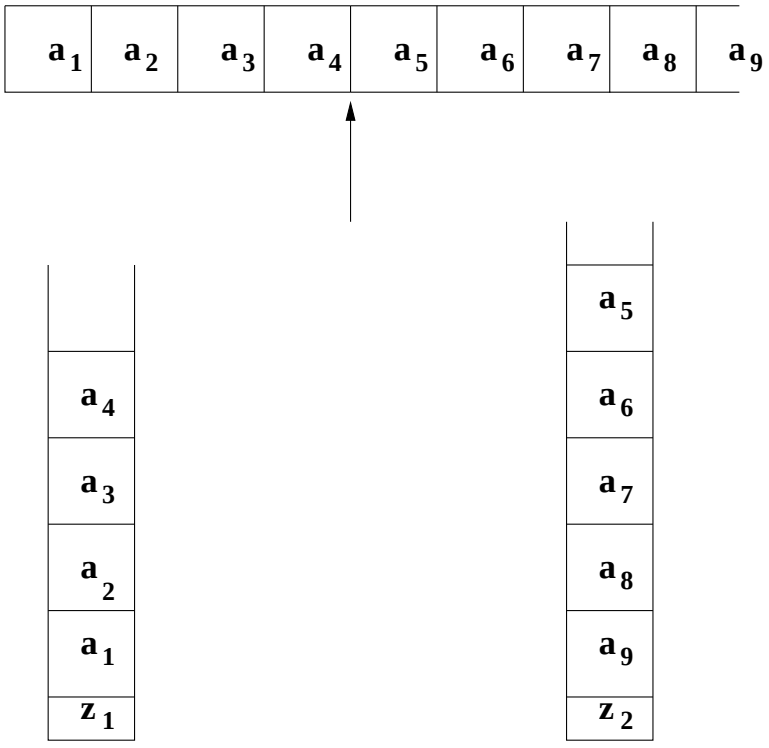
1. $(f_{i-1}, Z_{n+2}) \in \delta'_{n+2}(f_i, \lambda, Z_{n+2})$ for $2 \leq i \leq k$.

The set of final states F' is defined as

$$F' = \{f_1\} \cup \{p \in F \mid \exists q \ s.t \ \delta_i([q,k-1],a,X)$$
$$\text{contains } p \text{ for some } i, \ a \in V \text{ and } X \in \Gamma'_i\}$$

Equivalence of 2-PDA and TM

- The symbols simulated and which lies to the left of the head can be put in one stack whereas the symbols read but lies to the right of the head can be put in the second stack.
- In each of the stack the symbols closer to the head are placed closer to the top of the stack.



Acceptance by Empty store

- $N(M) = \{w \in V^* \mid (q_o, w, Z_1, Z_2, \cdots, Z_n, i') \vdash^* (q, \lambda, \lambda, \lambda, \cdots, \lambda(n\ times), i)\}$
- $L(M) \Longrightarrow N(M)$
  1. $Q'_i = Q_i \cup \{q_i\},\ 1 \leq i \leq n$
  2. $\Gamma'_i = \Gamma_i \cup \{Z'_i\}$
  3. For each i $\delta'_i(q, a, X)$ includes all elements of $\delta_i(q, a, X)$ $\forall q \in Q_i, a \in V, X \in \Gamma_i$
  4. $\delta'_i(q, \lambda, Z'_i)$ contains $(q, Z_i Z'_i)$
  5. For all $i (1 \leq i \leq n)$ if $q \in F \cap Q_i$ then $\delta'_i(q, \lambda, X) = (q_1, X)$
  6. For $1 \leq i \leq n-1$
     (a) $\delta'_i(q_i, \lambda, X)$ contains $(q_i, \lambda),\ X \in \Gamma_i$
     (b) $\delta'_i(q_i, \lambda, Z_i)$ contains (q_{i+1}, λ)
  7. $\delta'_n(q_n, \lambda, Z'_n)$ contains (q_n, λ)

Distributed Deterministic PDA: Definition

- An DDPDA is a 7-tuple $M = (Q, V, \Gamma, \Delta, q_0, Z, F)$ where,
- M is a n-PDA with the following:
- Δ is an n-tuple where each $\delta_i : Q_i \times (V \cup \{\lambda\}) \times \Gamma_i \longrightarrow Q_{union} \times \Gamma_i^*$,with the following restrictions:
  - for each $q \in Q_i$ and $Z \in \Gamma_i$, whenever $\delta_i(q, \epsilon, Z)$ is nonempty, then $\delta_i(q, a, Z)$ is empty for all $a \in V$;
  - for no $q \in Q_i$, $Z \in \Gamma_i$, and $a \in V \cup \{\epsilon\}$ does $\delta_i(q, a, Z)$ contain more than one element.

Important Note

- **Note** that in the previous defn. event hough every component is deterministic locally the choice of the next component is nondeterministic.
- Making the choice of the next component also deterministic decreases the power drastically. The system accepts only the languages accepted by a deterministic PDA which can not accept ww.Making the choice of the next component nondeterministic and each component deterministic locally does not decrease the power of the distributed PDA in the case of t-mode. In case of other modes the system requires three or four components to get the power of a TM.

Example: DDPDA

Consider the 4-DPDA M

- $Q_1 = \{q_0, q_a, q_b, q_T\}, Q_2 = \{q_{a'}, q_{b'}, q_s, q_f\}$
- $Q_3 = Q_4 = \{q'_T, q''_T\}, V = \{a, b\}, F = \{q_f\}$
- $\Gamma_1 = \Gamma_2 = \{Z_1, a, b\}, \Gamma_3 = \{Z_3\}, \Gamma_4 = \{Z_4\}$ Let $X \in \{Z_1, a, b\}$ and $Y \in \{Z_2, a, b\}$.
- $\delta_1(q_0, a, Z_1) = \{(q_a, aZ_1)\}$
- $\delta_1(q_0, b, Z_1) = \{(q_b, bZ_1)\}$
- $\delta_1(q_a, b, X) = \{(q_a, bX)\}$
- $\delta_1(q_b, a, X) = \{(q_b, aX)\}$
- $\delta_1(q_a, a, X) = \{(q'_T, X)\}$
- $\delta_1(q_b, b, X) = \{(q''_T, X)\}$
- $\delta_3(q'_T, \lambda, Z_3) = \{q_T, Z_3\}$
- $\delta_3(q''_T, \lambda, Z_3) = \{q_T, Z_3\}$

- $\delta_4(q'_T, \lambda, Z_4) = \{(q_a, Z_4)\}$
- $\delta_4(q''_T, \lambda, Z_4) = \{(q_b, Z_4)\}$
- $\delta_1(q_T, \lambda, a) = \{(q_{a'}, \lambda)\}$
- $\delta_1(q_T, \lambda, b) = \{(q_{b'}, \lambda)\}$
- $\delta_2(q_{a'}, \lambda, Y) = \{(q_T, aY)\}$
- $\delta_2(q_{b'}, \lambda, Y) = \{(q_T, bY)\}$
- $\delta_1(q_T, \lambda, Z_1) = \{(q_e, Z_1)\}$
- $\delta_2(q_e, \lambda, \{a, b\}) = \{(q_s, \lambda)\}$
- $\delta_2(q_s, a, a) = \{(q_s, \lambda)\}$
- $\delta_2(q_s, b, b) = \{(q_s, \lambda)\}$
- $\delta_2(q_s, \lambda, Z_2) = \{(q_f, Z_2)\}$
- The above 4-DPDA working in t-mode accepts the following language L:

$$L \;=\; \{ww \mid w \in \{a, b\}^*\}.$$

Equivalence of DCDPDA and Deterministic PDA

- $Q' = Q_{union}$
- $\Gamma' = \{[Y_1, Y_2, \cdots, Y_n] \mid Y_i \in \Gamma_i\}$
- $Z' = [Z_1, Z_2, \cdots, Z_n]$
- If the transition involves a push

$$\delta'(q, a, [X_1, X_2, \cdots, X_i, \cdots, X_n]) \ =$$

$$(p, [X_1, X_2, \cdots, \gamma X_i, \cdots, X_n])$$

$$\text{if } \delta_i(q, a, X_i) \ = \ (p, \gamma X_i)$$

- If the transition involves a pop

$$\delta'(q, a, [X_1, X_2, \cdots, X_i, \cdots, X_n]) \ =$$

$$(p, [X_1, X_2, \cdots, \lambda, \cdots, X_n])$$

$$\text{if } \delta_i(q, a, X) \ = \ (p, \lambda)$$

DDPDA: Proofs

- ***-mode $\Longrightarrow$ t-mode**
  - $Q'_i = Q_i$ for all $1 \leq i \leq n$
  - $Q'_{n+1} = \{q^* \mid q \in$ more than one component$\}$
  - If $\delta_i(p, a, X) = (q, \alpha)$ and q is not in more than one component then $\delta'_i(q, a, X) = \delta_i(q, a, X)$ for $q \in Q_i, a \in V, X \in \Gamma_i, 1 \leq i \leq n$.
  - for all q in more than one component, in which the component i is one, $\delta'_i(p, a, X) = (q^*, y)$ if $\delta_i(p, a, X) = (q, y), a \in V, X \in \Gamma_i$
  - $\delta'_{n+1}(q^*, \lambda, Z_{n+1}) = (q, Z_{n+1})$

Contd..

- **t-mode $\Longrightarrow$ *-mode**
  - if $q_0 \in Q_i$ then $Q'_i = \{[q,i] \mid 1 \leq i \leq n, q \in Q_i\} \cup \{q^* \mid q \in Q_i\} \cup \{q_0\}$.
  - if $q_0 \notin Q_i$ then $Q'_i = \{[q,i] \mid 1 \leq i \leq n, q \in Q_i\} \cup \{q^* \mid q \in Q_i\}$.
  - $\delta'_i(q_0, \lambda, X) = ([q_0, i], X)$
  - $\delta'_i([q,i], a, X) = ([p,i], \gamma X)$ if $\delta_i(q, a, X) = (p, \gamma X)$ and $p \in Q_i$.
  - $\delta'_i([q,i], a, X) = (p^*, \gamma X)$ if $\delta_i(q, a, X) = (p, \gamma X)$ and $p \notin Q_i$.
  - $\delta'_j(p^*, \lambda, X) = ([p,j], X)$
  - $F' = \{[q,i] \mid q \in F, 1 \leq i \leq n\}$

Contd..

- $= k$-**mode** $\Longrightarrow t$-**mode**

  1. $Q_i' = \{[q,t] \mid q \in Q_i \text{ and } 1 \leq t \leq k\} \cup \{q_j \mid q \in Q_i \cap Q_j,\ 1 \leq j \leq n \text{ and } i \neq j\}$
  2. $\delta_i'([p,l],a,X) = ([s,l{+}1],y)$ provided $\delta_i'(p,a,X) = (s,y)$ and $l < k$
  3. If $l = k$ then $\delta_i'([q,l],a,X) = (s_i,y)$ if $s \in Q_i \cap Q_j$ for $X \in \Gamma_i$ and $\delta_i(q,a,X) = (s,y)$
  4. $\delta_j'(q_i,\lambda,X) = ([q,1],X)$ for $X \in \Gamma_j$ and for $i,j (i \neq j)$ such that $q \in Q_i \cap Q_j$.
  5. $F' = \{[q,k] \mid q \in F\}$

Contd..

- **t-mode $\Longrightarrow=$ k-mode**

  1. $Q'_i = Q_i \cup \{[q,s] \mid 1 \leq s \leq k,\ q \in Q_i\} \cup \{r'_{qi} \mid q \in Q_i\} \cup \{r_{q'_s} \mid q \in Q_{T_i},\ 1 \leq i \leq n,\ q'$ is the state outside the domain of the component i which can be arrived from the state $q, 1 \leq s \leq k\} \cup \{p_{i_1}, p_{i_2}, \cdots, p_{i_k}\} \cup \{f_{q,1}, f_{q,2}, \cdots, f_{q,k} \mid q \in F \cap Q_i\} \cup \{p_{q,i,s,j} \mid 1 \leq i \leq n,\ 1 \leq s,j \leq k,\ q \in Q_i\}$ for $1 \leq i \leq n$.
  2. $Q'_{n+1} = \{r_{qi} \mid q \in Q_i, 1 \leq i \leq n\} \cup \{d_1, d_2, d_3, \cdots, d_k\}$.
  3. $Q'_{n+2} = \{r_{q,i,s,j} \mid 1 \leq i \leq n,\ 1 \leq s,j \leq k, q \in Q_{union}\}$
  4. $\Gamma'_i = (\Gamma_i - Z_i) \cup Z'_i \cup \{Y_i \mid 1 \leq i \leq k\}$ for $1 \leq i \leq n$.
  5. $\Gamma'_{n+1} = \{Z_{qi} \mid q \in Q_i, 1 \leq i \leq n\} \cup \{Z_{n+1}\}$.

6. $\Gamma'_{n+2} = \{Z'_{n+2}\}$

Let Q_{T_i} be the set of all states q in $Q - Q_i$ such that q is a transition state for the component i and let $N(Q_{T_i})$ be the set of all states q in Q_i such that from q the system can arrive at a state $p \in Q_{T_i}$ by a single transition.

1. $\delta'_i(q_0, a, X) = ([q, 1], \gamma X)$ if $\delta_i(q_0, a, X) = (q, \gamma X)\ \forall i$ s.t $q \in Q_i,\ 1 \leq i \leq n$.
2. $\delta'_i(q, \lambda, X) = ([q, 1], X)\ \forall i,\ 1 \leq i \leq n$.
3. $\delta'_i(r'_{qi}, \lambda, X) = ([q, 1], X)$.
4. $\delta'_i(r_{q'_s}, a, X) = (p_{i_{k-s-2}}, Z_{qi}X)$.
5. $\delta'_i(p_{i_j}, \lambda, Z_{qi}) = (p_{i_{j-1}}, Z_{qi})$ for $2 \leq j \leq k$.
6. $\delta_i(p_{i_1}, \lambda, Z_{qi}) = (q, \lambda)$.
7. $\delta'_i(p_{q,i,s,j}, \lambda, X) = (p_{q,i,s,j-1}, X) 1 < j \leq s$.
8. $\delta'_i([p_{q,i,s,1}, a, X) = ([r, s+1], X)$ if $\delta_i(q, a, X) = (r, X)$.

Case 1 Let $q \notin F$

1. if $q \in N(Q_{T_i})$ and $s < k-1$ then $\delta'_i([q,s],a,X) = (r_{q'_s}, \gamma X)$ if $\delta_i(q,a,X) = (q', \gamma X)$.
2. if $q \in N(Q_{T_i})$ and $s = k-1$ then $\delta'_i([q,s],a,X) = (p, \gamma X)$ if $\delta_i(q,a,X) = (p, \gamma X)$.
3. if $q \notin N(Q_{T_i})$ and $s < k-1$ then $\delta'_i([q,s],a,X) = ([p,s+1], \gamma X)$ if $\delta_i(q,a,X) = (p, \gamma X)$.
4. if $q \notin N(Q_{T_i})$ and $s = k-1$ then $\delta_i([q,s],a,X) = (r_{qi}, X)$.

Case 2 Let $q \in F$

1. if $s \leq k-1$ then $\delta'_i([q,s],a,X) = (f_{q,k-s-1}, Y_s X)$
2. $\delta'_{n+1}(f_{q,k-j}, \lambda, Y_s)) = (f_{q,k-j-1}, Y_s)$
3. $\delta'_{n+1}(f_{q,1}, Y_s) = (r_{q,i,s,1}, \lambda)$

other δ'_{n+1} definitions are,

1. $\delta'_{n+1}(r_{qi}, \lambda, Z_{n+1}) = (d_2, Z_{qi} Z_{n+1})$.

2. $\delta'_{n+1}(d_i, \lambda, Y) \ = \ (d_{i+1}, Y)$ for $2 \leq i \leq k-1$.

3. $\delta'_{n+1}(d_k, \lambda, Z_{qi}Z_{n+1}) \ = \ (r'_{qi}, Z_{n+1})$.

δ'_{n+2} is defined as,

1. $\delta'_{n+2}(r_{q,i,s,j}, \lambda, Z'_{n+2}) \ = \ (r_{q,i,s,j+1}, Z'_{n+2})$, $1 \leq j < k$.

2. $\delta'_{n+2}(r_{q,i,s,k}, \lambda, Z'_{n+2}) \ = \ (p_{q,i,s,s}, Z'_{n+2})$.

The set of final states F' is defined as

$$F' \ = \ \{f_1\}$$

Watson-Crick DFSA: Definitions

- A distributed Watson-Crick finite state automata is a construct $M = (V, \rho, K, s_0, F, \Delta)$ where

  1. V is a finite set of symbols called alphabets
  2. ρ is a symmetrical relation on V
  3. K is a n-tuple $(K_1, K_2, \cdots, K_n)$ where each K_i is a finite set of states
  4. $s_0 \in \cup_i K_i$ is the start state
  5. $F \subseteq \cup_i K_i$ is the set of final states
  6. Δ is a n-tuple $(\delta_1, \delta_2, \cdots, \delta_n)$ where each $\delta_i : K_i \times \begin{pmatrix} V^* \\ V^* \end{pmatrix} \longrightarrow 2^{\cup_i K_i}$ is a mapping called transition mapping of the component i

Contd..

- The transitions of M are written in the following form, $s\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \Longrightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} s'$ This has the same meaning as $s' \in \delta(s, \begin{pmatrix} x_1 \\ x_2 \end{pmatrix})$
- **<u>t-mode</u>**

  $$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}(s,i)\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \Longrightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}(s',i)\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

  iff

  $$s' \in \delta_i(s, \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}) \text{ and } s, s' \in K_i$$

  if $s' \neq K_i$ then we write

  $$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}(s,i)\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \Longrightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}(s',j)\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

  provided $s' \in K_j (j \neq i)$

Contd..

- $\underline{= k\textbf{-mode}}$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} (s, i, l) \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \Longrightarrow$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} (s', i, l+1) \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

if

$$s' \in \delta_i(s, \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}), l < k, \text{ and } s, s' \in K_i$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} (s, i, k) \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} \Longrightarrow$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} (s, j, 0) \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

if

$$s' \in \delta_i(s, \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}) \text{ and } s' \in K_j$$

Contd..

- The language accepted by the distributed watson-crick finite state automata $M = (V, \rho, K, s_0, F, \Delta)$ in the acceptance mode $f \in D$ is

$$L_f(M) = \{w_1 \in V^* \mid s_0 \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \Longrightarrow^* \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} s_f,$$

for $s_f \in F$, and $w_2 \in V^*$, $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V)\}$

Variants

- We can also consider several variants of Watson-Crick finite automata. We say that $M = (V, \rho, K, s_0, F, \Delta)$ is

  - *stateless,* if $\cup_i K_i = F = \{s_0\}$;

  - *all-final,* if $F = K$,

  - *simple,* if for all $s \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \longrightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} s' \in \delta_i$ we have either $x_1 = \lambda$ or $x_2 = \lambda$, $1 \leq i \leq n$,

  - *1-limited,* if for all $s \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \longrightarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} s' \in \delta_i$ we have $\mid x_1 x_2 \mid = 1$, $1 \leq i \leq n$.

Notations

- $DAWK_f(u)$ – distributed and arbitrary.
- $DNWK_f(u)$ – distributed and no state.
- $DFWK_f(u)$ – distributed and all-final.
- $DSWK_f(u)$ – distributed and simple.
- $D1WK_f(u)$ – distributed and 1-limited.
- $DNSWK_f(u)$ – distributed, no state and simple.
- $DN1WK_f(u)$ – distributed, no state and 1-limited.
- $DFSWK_f(u)$ – distributed, all-final and simple.
- $DF1WK_f(u)$ – distributed, all-final and 1-limited.

Results

- $N1WK \subset DN1WK_s(u)$ where $s \in \{t, = k, \leq k, \geq k\}$. $DN1WK_*(u) = N1WK$
  - t-**mode** $L_1 = a^+ \cup b^+$. (If $L \in NWK(u)$, then $L = L^+$. If $L \in N1WK(u)$, then there is an alphabet V such that $L = V^+$.)
  - $= k$-**mode** $L_2 = (ab)^*a + (ba)^*b$.
  - $\leq k$-**mode** L_3 where L_3 is the language over the alphabet $\{a, b\}$ in such a way that every continues occurrence of a and continues occurrence of b does not exceed k.

Contd..

- $NWK \subset DNWK_s(u)$ where $s \in \{t, = k, \leq k, \geq k\}$.

  $DNWK_*(u) = NWK$

  - t-**mode** L_1 holds good.
  - $= k$-**mode** the set of strings which satisfy the following: every string in the language if cut into some n substrings each of length k then each substring will be a string only over $\{a, b\}$ or $\{c, d\}$ not both. That too, if we label them in the order of occurrence of the substrings in the original string then they will occur alternatively.

– $\leq k$-**mode** the language which satisfies the following conditions:There exists numbers $k_1, k_2, \cdots, k_n$ where $k_i \leq k (1 \leq i \leq n)$ for every string in the language such that if the string is split into n substrings $\alpha_1, \alpha_2 \cdots, \alpha_n$ each of length k_i respectively then they will be either a string over $\{a, b\}$ or $\{c, d\}$ but not both as well as if $\alpha_i \in \{a, b\}^{k_i}$ then $\alpha_{i+1} \in \{c, d\}^{k_{i+1}}$ and vice-versa.

Watson-Crick PDA

- V_1 is a finite set of symbols called input alphabets
- $\rho_1 \subseteq V_1 \times V_1$ is a symmetrical relation on V_1
- V_2 is the pushdown alphabet
- $\rho_1 \subseteq V_2 \times V_2$ is a symmetrical relation on V_2
- K is the finite set of states, $s_0 \in K$ is the start state, $F \subseteq K$ is the set of final states
- $Z_0 = (Z_{01}, Z_{02}) \in V_2 \times V_2$ is the initial pushdown symbol
- $\delta : K \times \begin{pmatrix} V_1^* \\ V_1^* \end{pmatrix} \times \begin{pmatrix} V_2 \\ V_2 \end{pmatrix}$ into finite subsets of $K \times \begin{pmatrix} V_2^* \\ V_2^* \end{pmatrix}$ is a mapping called transition mapping

Contd..

- The language accepted by the Watson-Crick push-down automata

$M = (V_1, \rho_1, V_2, \rho_2, K, \Gamma, s_0, Z, F, \delta)$ is

$$L(M) = \{w_1 \in V_1^* \mid (s_0, \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}, \begin{pmatrix} Z_{01} \\ Z_{02} \end{pmatrix}) \vdash^*$$

$$(s_f, \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix}) \text{ for } s_f \in F,$$

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \in WK_\rho(V), \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \in \begin{pmatrix} V_2^* \\ V_2^* \end{pmatrix}\}$$

Equivalence of WKPDA and TM

$K' = \{(q,1) \mid q \in K_1\} \cup \{(q,2) \mid q \in K_2\} \cup \{s_0'\}, V_1 = V, \rho_1 = \{(a,a) \mid a \in V\}, V_2 = \Gamma_1 \cup \Gamma_2, \rho_2 = V_2 \times V_2, F' = \{(q,i) \mid q \in F, i = 1,2\}$

- $((s_0,i), \begin{pmatrix} Z_{01} \\ Z_{02} \end{pmatrix}) \in \delta'(s_0', \begin{pmatrix} \lambda \\ \lambda \end{pmatrix}, \begin{pmatrix} Z_{01} \\ Z_{02} \end{pmatrix})$
- $((q',1), \begin{pmatrix} YX_1 \\ X_2 \end{pmatrix}) \in \delta'((q,i), \begin{pmatrix} a \\ a \end{pmatrix}, \begin{pmatrix} X_1 \\ X_2 \end{pmatrix})$ if $(q', YX_1) \in \delta_1(q,a,X_1)$

  $((q',2), \begin{pmatrix} X_1 \\ YX_2 \end{pmatrix}) \in \delta'((q,i), \begin{pmatrix} a \\ a \end{pmatrix}, \begin{pmatrix} X_1 \\ X_2 \end{pmatrix})$ if $(q', YX_2) \in \delta_2(q,a,X_2)$ where $q, q' \in K_i$.
- If $q' \notin K_i$, $((q', i \ mod \ 2 + 1, \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}) \in$

  $\delta'((q,i), \begin{pmatrix} a \\ a \end{pmatrix}, \begin{pmatrix} X_1 \\ X_2 \end{pmatrix})$

Summary

- Defined Distributed FSA and Distributed PDA and considered the languages accepted by in all the well known modes of acceptance.
- We proved
  - Distributed FSA accepts only the REG.
  - In case of Distributed PDA all modes are equivalent. 2-PDA in all modes accept RE.
  - Acceptance by **Final State** is equivalent to acceptance by **Empty store**.
  - DDPDA is equivalent in all modes.In case of t-mode and *-mode 2-DPDA accepts RE. In case of other modes 4-DPDA accepts RE.
  - DCDPDA is equivalent to Deterministic PDA.

Summary: Contd..

- Defined Distributed Watson-Crick FSA and Watson-Crick PDA and considered the languages accepted by them
- We proved
  - The accepting power does not increase in all the variants of Distributed Watson-Crick FSA except for the Distributed No State Watson-Crick FSA.
  - Distributed Watson-Crick PDA in the arbitrary sense is as powerful as a Turing Machine.

Conclusions

- We considered only the Cooperating Distributed Machine models. It should be interesting to study the Parallel Communicating Machine models.
- Though our model for the Distributed Pushdown Automata seems to be quite natural and simple it is not equivalent to the CD Grammar Systems with Context Free rules. So, it should be interesting to find an equivalent machine model for that, even though it seems to be quite artificial to do that.
- In these models we have not looked in to the complexity point of view. It will be nice to explore ideas of improving the complexity of the machines proposed here.

- Watson-Crick models uses only the power of Complementarity in DNA computing. It is still an open problem to model the other feature of the DNA Computing, namely, Massive Parallelism.

Publications

1. K. Krithivasan, **M. Sakthi Balan** and P. Harsha, Distributed Processing in Automata, *International Journal of Foundations of Computer Science*, Vol. 10, No. 4, 443-464, 1999.

2. K. Krithivasan and **M. Sakthi Balan**, Distributed Processing in Deterministic PDA, accepted for presentation in *International Workshop on Grammar Systems*, Austria, July 3-7, 2000.

3. K. Krithivasan, **M. Sakthi Balan** and R. Rama, Array Contextual Grammars, *Commemorative Volume for S. Marcus* ed. Gh. Păun, accepted for publication.

4. K. Krithivasan and **M. Sakthi Balan**, Some properties of Array Contexual Grammars, presented in *National Seminar on Discreate Mathematics and Applications*, MS university, Tirunelveli, Jan 5-7, 2000.