

Linear Conjunctive Grammars and One-turn Synchronized Alternating Pushdown Automata

Tamar Aizikowitz* and Michael Kaminski
Technion – Israel Institute of Technology

Formal Grammar 2009, Bordeaux

Introduction



- Context-free languages combine expressiveness with polynomial parsing, making them appealing for practical applications.
- Several fields (*e.g.*, Programming Languages, Computational Linguistics, Computational Biology) have expressed a need for a **slightly stronger** language class.

Conjunctive Grammars and SAPDA

- Conjunctive Grammars (CG) [Okhotin, 2001] are an extension of context-free grammars.
 - ▣ Have explicit intersection rules.
 - ▣ Retain polynomial parsing \Rightarrow practical applications.
- Synchronized Alternating Pushdown Automata (SAPDA) [Aizikowitz *et al.*, 2008] extend PDA.
 - ▣ First automaton counterpart for Conjunctive Grammars.

Linear CG and One-turn SAPDA

- Linear Conjunctive Grammars (LCG) [Okhotin, 2001] are a sub-family of Conjunctive Grammars.
 - ▣ Analogous to Linear Grammars as a sub-family of Context-free Grammars.
 - ▣ Have efficient parsing algorithms [Okhotin, 2003]
 - ▣ Equivalent to Trellis Automata [Okhotin, 2004]
 - ▣ Related to Mildly Context Sensitive Languages
- We define a sub-family of SAPDA, **one-turn SAPDA**, and prove its equivalence to LCG.

Outline



- Model Definitions
 - ▣ Conjunctive Grammars
 - ▣ Linear Conjunctive Grammars
 - ▣ Synchronized Alternating Pushdown Automata
- Main Results
 - ▣ One-turn Synchronized Alternating PDA
 - ▣ Equivalence Results
- Linear Conjunctive Languages
 - ▣ Characterization of Language Class
 - ▣ Mildly Context Sensitive Languages



Model Definitions

Conjunctive Grammars

Linear Conjunctive Grammars

Synchronized Alternating Pushdown Automata

Conjunctive Grammars

- A **CG** is a quadruple: $G=(V, \Sigma, P, S)$

non-terminals terminals derivation rules start symbol

- **Rules:** $A \rightarrow (\alpha_1 \& \dots \& \alpha_n)$ s.t. $A \in V, \alpha_i \in (V \cup \Sigma)^*$

- **Derivation steps:**

$n=1 \Rightarrow$ standard CFG

- $s_1 A s_2 \Rightarrow s_1 (\alpha_1 \& \dots \& \alpha_n) s_2$ where $A \rightarrow (\alpha_1 \& \dots \& \alpha_n) \in P$

- $s_1 (w \& \dots \& w) s_2 \Rightarrow s_1 w s_2$ where $w \in \Sigma^*$

- **Language:** $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$

- **Note:** $(A \& B) \Rightarrow^* w$ iff $A \Rightarrow^* w \wedge B \Rightarrow^* w$

Linear Conjunctive Grammars

- A CG $G=(V, \Sigma, P, S)$ is **linear** if all rules are:
 - $A \rightarrow (u_1 B_1 v_1 \ \& \cdots \ \& \ u_n B_n v_n)$ s.t. $u_i, v_i \in \Sigma^*, B_i \in V$
 - $A \rightarrow w$ s.t. $w \in \Sigma^*$
- **Example:** a Linear CG for the **multiple-agreement** language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$:
 - $C \rightarrow Cc \mid D ; D \rightarrow aDb \mid \varepsilon \Rightarrow L(C) = \{a^n b^n c^m \mid n, m \in \mathbb{N}\}$
 - $A \rightarrow aA \mid E ; E \rightarrow bEc \mid \varepsilon \Rightarrow L(A) = \{a^m b^n c^n \mid n, m \in \mathbb{N}\}$
 - $S \rightarrow (C \ \& \ A) \Rightarrow L(S) = L(C) \cap L(A)$
 - $S \Rightarrow (C \ \& \ A) \Rightarrow (Cc \ \& \ A) \Rightarrow (Dc \ \& \ A) \Rightarrow (aDbc \ \& \ A)$
 \Rightarrow

Synchronized Alternating Pushdown Automata

- Synchronized Alternating Pushdown Automata (SAPDA) are an extension of classical PDA.
- **Transitions** are made to **conjunctions** of (state , stack-word) pairs, *e.g.*,

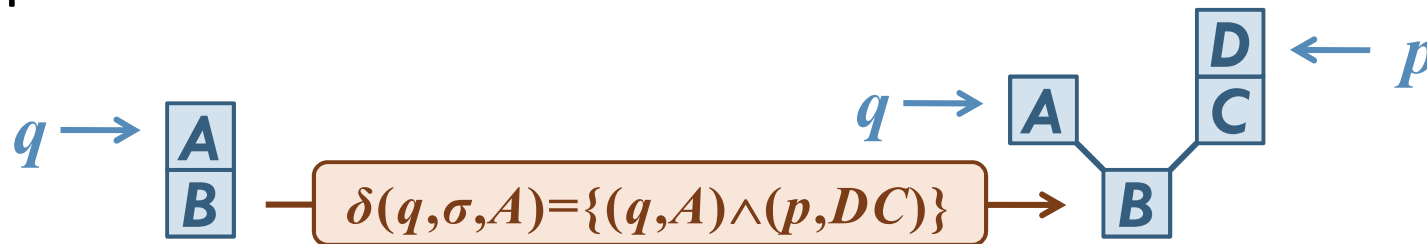
$$\delta(q, \sigma, X) = \{ (p_1, XX) \wedge (p_2, Y), (p_3, Z) \}$$

non-deterministic model = many possible transitions

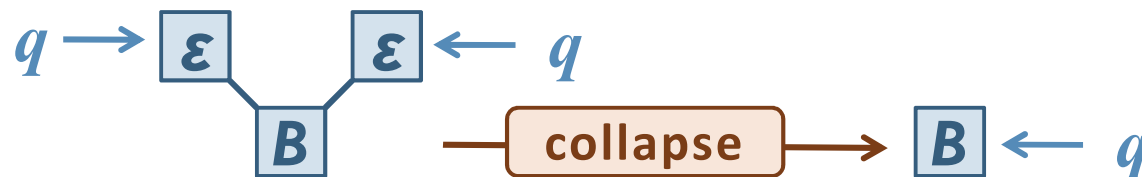
- **Note:** if all conjunctions are of **one** pair only, the automaton is a “regular” PDA.

SAPDA Stack Tree

- The stack of an SAPDA is a **tree**. A transition to n pairs splits the current branch into n branches.

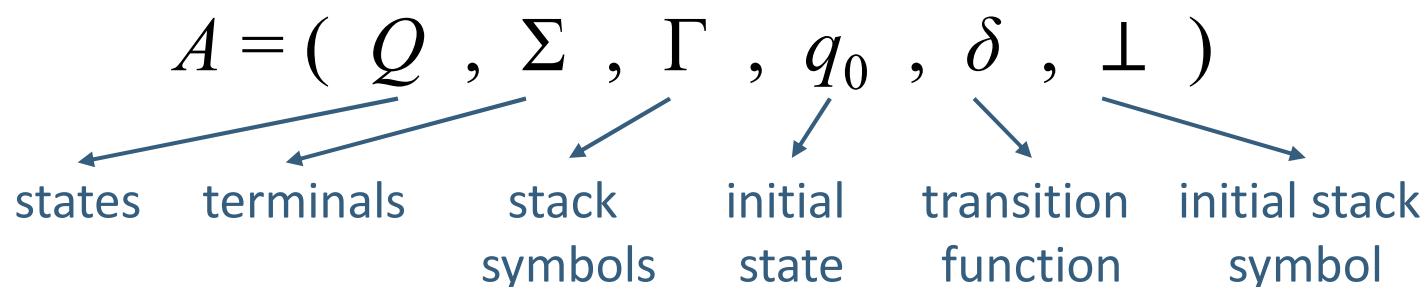


- Branches are processed **independently**.
- Empty sibling branches can be collapsed if they are **synchronized** = are in the same state and have read the same portion of the input.



SAPDA Formal Definition

- An **SAPDA** is a sextuple



- **Transition function:**

$$\delta(q, \sigma, X) \subseteq \{ (q_1, \alpha_1) \wedge \cdots \wedge (q_n, \alpha_n) \mid q_i \in Q, \alpha_i \in \Gamma^*, n \in \mathbb{N} \}$$

SAPDA Computation and Language

□ Computation:

- Each step, a transition is applied to one stack-branch
- If a stack-branch is empty, it cannot be selected
- **Synchronous** empty sibling branches are collapsed

have the same state and remaining input

□ **Initial Configuration:** $\perp \leftarrow q_0$

□ **Accepting Configuration:** $\varepsilon \leftarrow q$

□ **Language:** $L(A) = \{w \in \Sigma^* \mid \exists q \in Q, (q_0, w, \perp) \vdash^* (q, \varepsilon, \varepsilon)\}$

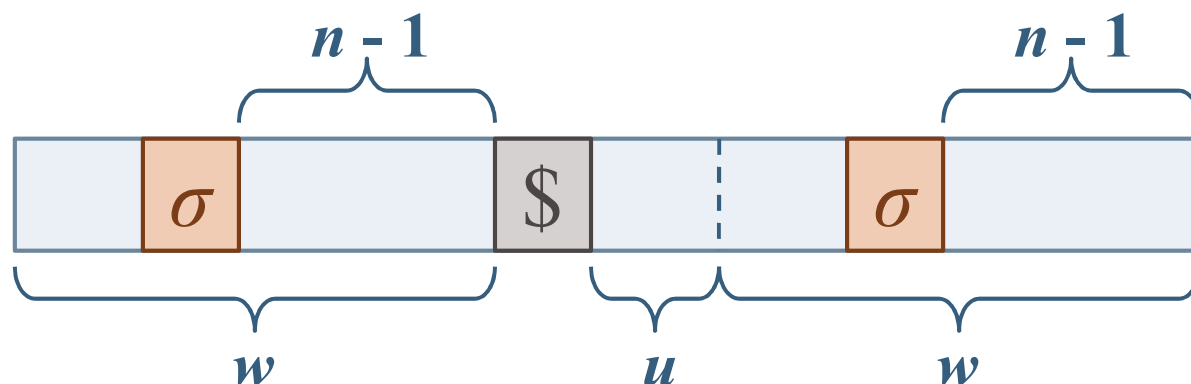
□ **Note:** all branches must empty \sim must “agree”.

Reduplication with a Center Marker

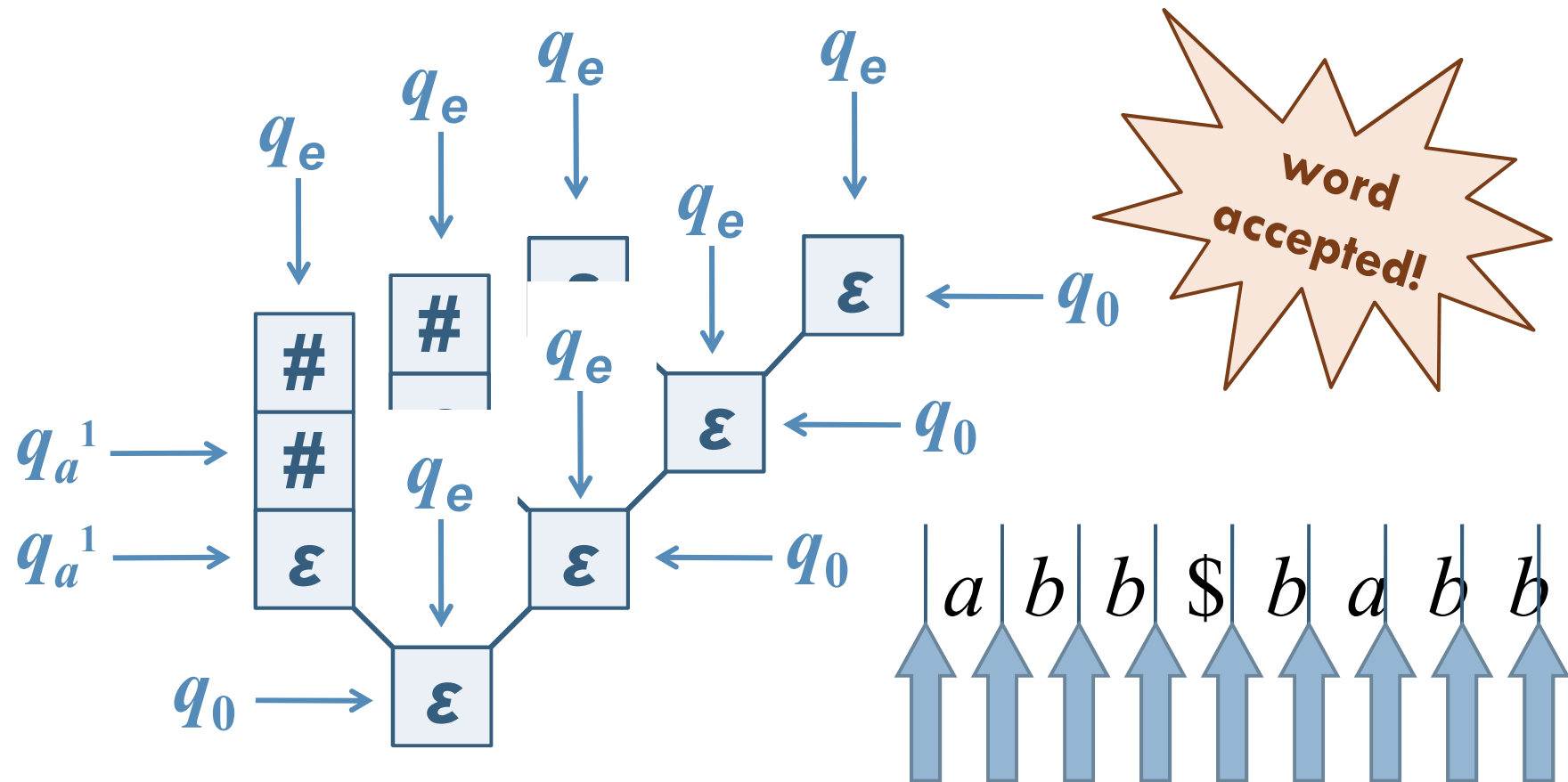
- The **reduplication with a center marker** language (RCM), $\{ w\$w \mid w \in \Sigma^* \}$, describes structures in various fields, *e.g.*,
 - **Copying phenomena in natural languages:**
“deal or no deal”, “boys will be boys”,
“is she beautiful or is she beautiful?”
 - **Biology:** microRNA patterns in DNA, tandem repeats
- We will construct an SAPDA for RCM.
- **Note:** it is not known whether **reduplication without a center marker** can be derived by a CG.

Example: SAPDA for RCM

- We consider an SAPDA for $\{w\$uw \mid w, u \in \Sigma^*\}$, which can easily be modified to accept RCM.
- The SAPDA is especially interesting, as it utilizes **recursive conjunctive transitions**.
- **Construction Idea:** if σ in the n^{th} letter before the \$, check that the n^{th} letter from the end is also σ .



Computation of SAPDA for RCM



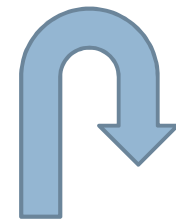
Main Results

One-turn Synchronized Alternating PDA

Equivalence Results

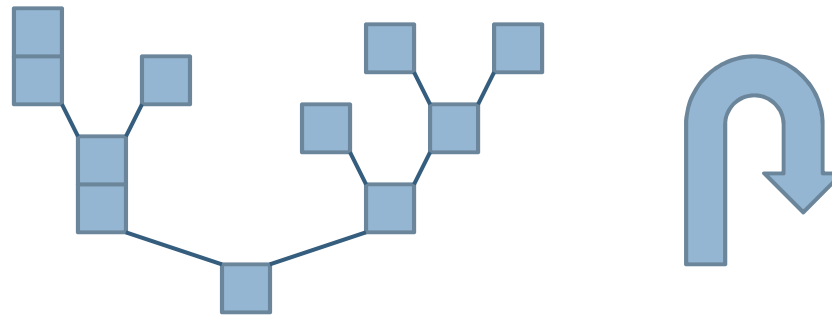
Motivation

- It is a well known result [Ginsburg *et al.*, 1966] that Linear Grammars are equivalent to one-turn PDA.
- A **one-turn PDA** is a PDA s.t. all accepting computations, have only one turn.
- A **turn** is a computation step where the stack height changes from increasing to decreasing.



One-turn SAPDA

- We introduce a similar notion of **one-turn SAPDA**, analogously to one-turn PDA.
- An SAPDA is **one-turn** if all stack-branches make exactly one turn in all accepting computations.

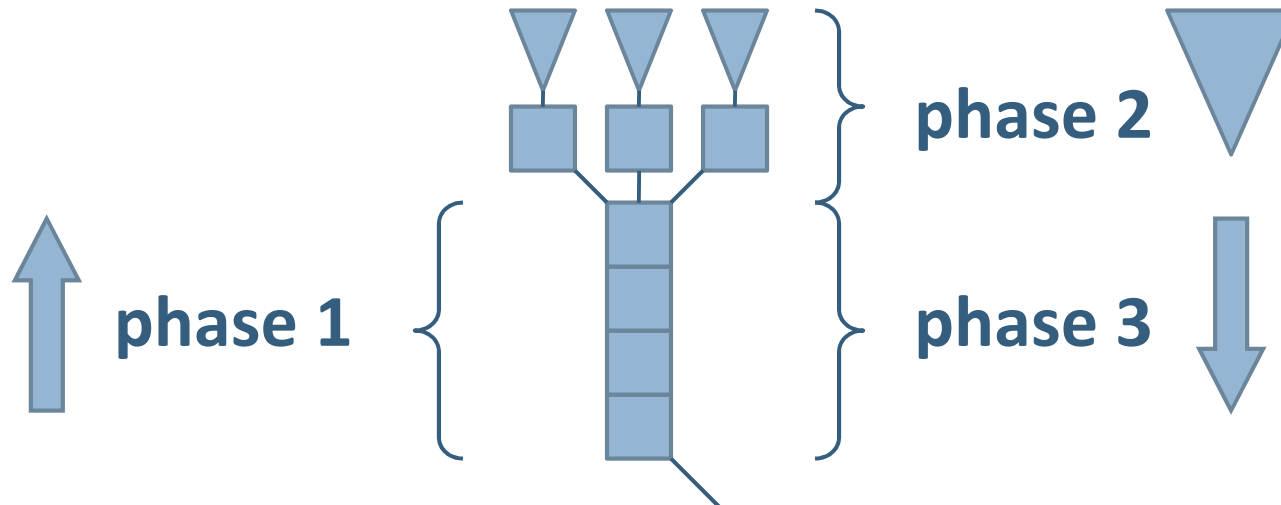


- **Note:** the requirement of a turn is not limiting as we are considering acceptance by empty stack.

Informal Definition

- Assume all transitions on a stack-branch and its sub-tree are applied consecutively (reordering if needed).
- We refer to this segment of the computation as the **relevant transitions** w.r.t. the branch.
- An SAPDA is **one-turn** if for every branch, the relevant transitions can be split into three phases:
 - (1) Increasing transitions applied to the stack-branch.
 - (2) A conjunctive transition followed by transitions applied to the branches in the sub-tree, and then a collapsing transition of the sub-tree.
 - (3) Decreasing transitions on the stack-branch.

Informal Definition *Continued...*



- **Note:** if the automaton is a classical PDA, then there is only one branch with no second phase (no conjunctive transitions), and therefore the automaton is a classical one-turn PDA.
- **Example:** The SAPDA we saw for RCM is one-turn.

Equivalence Results

- **Theorem 1.** *A language is generated by an LCG if and only if it is accepted by a one-turn SAPDA.*
- This result **mirrors the classical equivalence** between Linear Grammars and one-turn PDA, strengthening the claim of SAPDA as a **natural automaton counterpart** for CG.
- **Corollary:** One-turn SAPDA are equivalent to Trellis automata.

Proof Sketch

- **“only if”**: given an Linear Conjunctive Grammar, we construct a one-turn SAPDA.
 - ▣ Extension of the classical construction of a Context-free Grammar from a PDA.
- **“if”**: given a one-turn SAPDA we construct an equivalent Linear Conjunctive Grammar.
 - ▣ Extension of the classical construction of a LG from a one-turn PDA [Ginsburg *et al.*, 1966].
- The **full proofs** for both directions can be found in [Aizikowitz *et al.*, Technical Report CS-2009-15, Technion].



Linear Conjunctive Languages

Characterization of Language Class

Mildly Context Sensitive Languages

Characterization of Language Class

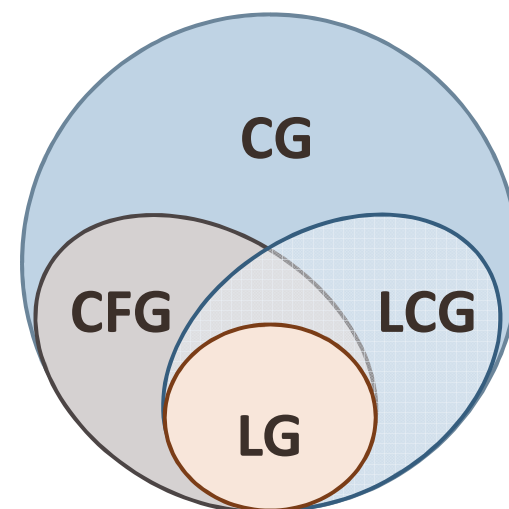
- LCG can derive all finite conjunctions of linear CF languages as well as some additional languages (e.g., RCM).

- **Closure Properties:**

- Union, intersection, complement, inverse homomorphism ✓
- Concatenation, ϵ -free homomorphism ✗
- Kleene star ?

- **Decidability Problems:**

- Membership: $O(n^2)$ time and $O(n)$ space. ✓
- Emptiness, finiteness, equivalence, inclusion, regularity ✗



Mildly Context Sensitive Languages

- We explore the correlation between LC Languages and **Mildly Context Sensitive Languages** (MCSL).
- Consider the categorization of MCSL:
 - (1) They contain the context-free languages ✗
⇒ this requirement holds only for general CG.
 - (2) They contain multiple-agreement, cross-agreement and reduplication ✓
 - (3) They are polynomially parsable ✓
 - (4) They are semi-linear ✗

Summary



- One-turn SAPDA introduced as a sub-family of Synchronized Alternating Pushdown Automata.
- One-turn SAPDA shown to be equivalent to Linear Conjunctive Grammars.
- The result strengthens SAPDA's claim as a natural automaton counterpart for Conjunctive Grammars.
- Conjunctive Grammars are a very appealing model for potential practical application, and thus encourage further research.



Thank you.

References

- Aizikowitz, T., Kaminski, M.: **Conjunctive grammars and alternating pushdown automata**. *WoLLIC'09*. LNAI 5110 (2008) 30 – 41
- Aizikowitz, T., Kaminski, M.: **Linear conjunctive grammars and one-turn synchronized alternating pushdown automata**. *Technical Report CS-2009-15*. Technion (2009)
- Ginsburg, S., Spanier, E.h.: **Finite-turn pushdown automata**. *SIAM Journal on Control*. 4(3) (1966) 429 – 453
- Okhotin, A.: **Conjunctive grammars**. *Journal of Automata, Languages and Combinatorics*. 6(4) (2001) 519 – 535
- Okhotin, A.: **Efficient automaton-based recognition for linear conjunctive languages**. *IJFCS*. 14(6) (2003) 1103 – 1116
- Okhotin, A.: **On the equivalence of linear conjunctive grammars and trellis automata**. *RAIRO Theoretical Informatics and Applications*. 38(1) (2004) 69 – 88

Construction of SAPDA for RCM

$$A = (Q, \{a, b, \$\}, \{\perp, \#\}, q_0, \delta, \perp)$$

$$Q = \{q_0, q_e\} \cup \{q_\sigma^i \mid \sigma \in \{a, b\}, i \in \{1, 2\}\}$$

$$\delta(q_0, \sigma, \perp) = \{(q_\sigma^1, \perp) \wedge (q_0, \perp)\}$$

recursively open branch
to verify σ is n^{th} from the
\$ and from the end

$$\delta(q_\sigma^1, \tau, X) = \{(q_\sigma^1, \#X)\}$$

"count" symbols between σ and \$

$$\delta(q_0, \$, \perp) = \{(q_e, \varepsilon)\}$$

\$ reached, stop recursion

$$\delta(q_\sigma^1, \$, X) = \{(q_\sigma^2, X)\}$$

\$ reached, stop "counting" symbols

$$\delta(q_\sigma^2, \tau \neq \sigma, X) = \{(q_\sigma^2, X)\}$$

look for σ

"guess" that this is the
 σ we are looking for,
or keep looking

$$\delta(q_\sigma^2, \sigma, X) = \{(q_e, X), (q_\sigma^2, X)\}$$

$$\delta(q_e, \tau, \#) = \{(q_e, \varepsilon)\}$$

"count" symbols from σ to the end

$$\delta(q_e, \varepsilon, \perp) = \{(q_e, \varepsilon)\}$$

all done: empty stack

“only if” Proof Sketch *Continued...*

$$\begin{aligned}
 S &\Rightarrow^* w_1 A w_2 \\
 &\Downarrow \\
 &w_1(u_1 B_1 v_1 \&\dots)w_2 \\
 &\Downarrow \\
 &\dots u_1 x C y v_1 \dots \\
 &\Downarrow \\
 &\dots u_1 x(\dots \&\dots) y v_1 \dots \\
 &\Downarrow^* \\
 &\dots u_1 x w y v_1 \dots
 \end{aligned}$$

