Conjunctive Grammars and Synchronized Alternating Pushdown Automata

Tamar Aizikowitz

Joint work with Michael Kaminski

Technion – Israel Institute of Technology

October 2009

Context-Free Languages

- Combine expressiveness with polynomial parsing
  $\Rightarrow$ appealing for practical applications.
- Possibly the most widely used language class in Computer Science.
- At the theoretical basis of Programming Languages, Computational Linguistics, Formal Verification, Computational Biology, and more.

Extended Models

- **Goal:** Models of computation that generate a **slightly stronger** language class without sacrificing polynomial parsing.
- **Why?** Such models seem to have great potential for practical applications.
- In fact, several fields (*e.g.,* Computational Linguistics) have already voiced their need for a stronger language class.

Conjunctive Grammars

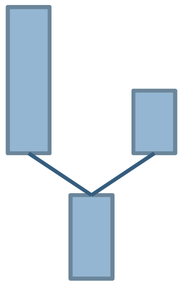
- Conjunctive Grammars (CG) [Okhotin, 2001] are an extension of context-free grammars.
- Have **explicit intersection** rules

$$S \Rightarrow (A \,\&\, B) \Rightarrow \cdots \Rightarrow (w \,\&\, w) \Rightarrow w$$

- **Semantics:** $L(A \,\&\, B) = L(A) \cap L(B)$
- **Recall:** Context-free languages are not closed under intersection $\Rightarrow$ **stronger language class**
- Retain **polynomial parsing** $\Rightarrow$ practical applications

Synchronized Alternating PDA

- Synchronized Alternating Pushdown Automata (SAPDA) [Aizikowitz Kaminski, 2008] extend PDA.
- Stack modeled as tree $\Rightarrow$ all braches must accept
- Uses a limited form of synchronization to create localized parallel computations.
- **First** automaton counterpart shown for Conjunctive Grammars.



- One-turn SAPDA shown to be equivalent to Linear CG [Aizikowitz Kaminski, 2009], mirroring the classical equivalence between one-turn PDA and LG.

Outline

- Model Definitions
  - Conjunctive Grammars
  - Synchronized Alternating Pushdown Automata (SAPDA)
- Main Results
  - Equivalence Results
  - Linear Conjunctive Grammars and One-turn SAPDA
- Conjunctive Languages
  - Characterization of Language Class
  - A Simple Programming Language
  - Mildly Context Sensitive Languages
- Summary and Future Directions

Model Definitions

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

  $n=1 \Rightarrow$ **standard CFG**

- **Examples:** $A \rightarrow (aAB \;\&\; Bc \;\&\; aD) \quad ; \quad A \rightarrow abC$

- **Derivation steps:**
  - $s_1 A s_2 \Rightarrow s_1(\alpha_1 \,\&\, \cdots \,\&\, \alpha_n)s_2$ where $A \rightarrow (\alpha_1 \,\&\, \cdots \,\&\, \alpha_n) \in P$
  - $s_1(w \,\&\, \cdots \,\&\, w)s_2 \Rightarrow s_1\, w\, s_2$ where $w \in \Sigma^*$

Grammar Language

- **Language:** $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$
- **Informally:** All terminal words w derivable from the start symbol S.
- **Note:** As (,) , and & are not terminal symbols, all conjunctions must be collapsed in order to derive a terminal word.
- **Semantics:** $(A \,\&\, B) \Rightarrow^* w$ iff $A \Rightarrow^* w \;\wedge\; B \Rightarrow^* w$

  Therefore, $L(A \,\&\, B) = L(A) \cap L(B)$.

Example: Multiple Agreement

- **Example:** following is a CG for the **multiple-agreement** language $\{a^n b^n c^n \mid n \in \mathbb{N}\}$:
  - $C \to Cc \mid D \;\; ; \;\; D \to aDb \mid \varepsilon \quad \Rightarrow L(C) = \{a^n b^n c^m \mid n,m \in \mathbb{N}\}$
  - $A \to aA \mid E \;\; ; \;\; E \to bEc \mid \varepsilon \quad \Rightarrow L(A) = \{a^m b^n c^n \mid n,m \in \mathbb{N}\}$
  - $S \to (C \,\&\, A) \qquad \Rightarrow L(S) = L(C) \cap L(A)$
  - $\boldsymbol{S} \Rightarrow (\boldsymbol{C} \,\&\, A) \Rightarrow (\boldsymbol{C}c \,\&\, A) \Rightarrow (\boldsymbol{D}c \,\&\, A) \Rightarrow (a\boldsymbol{D}bc \,\&\, A) \Rightarrow$

    $\Rightarrow (abc \,\&\, \boldsymbol{A}) \Rightarrow \cdots \Rightarrow (abc \,\boldsymbol{\&}\, abc) \Rightarrow abc$

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.*,

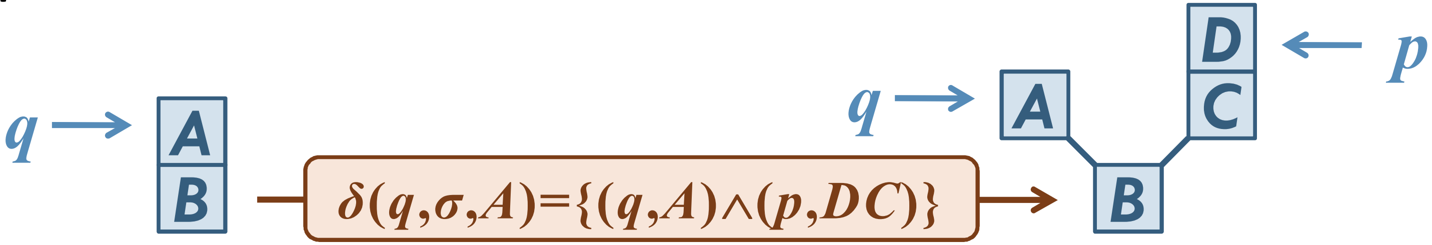
$$\delta(q\,,\sigma\,,X) = \{\,(p_1\,,XX)\wedge(p_2\,,Y)\ \ ,\ \ (p_3\,,Z)\,\}$$

non-deterministic model = many possible transitions

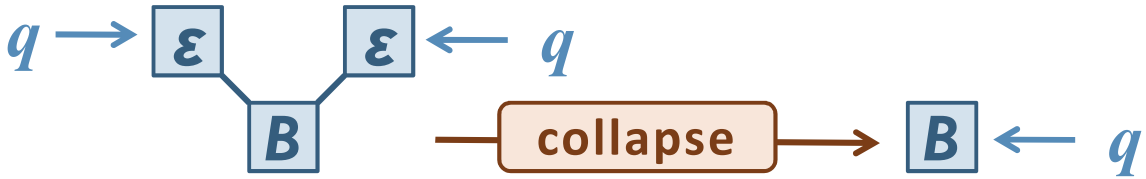
- **Note:** if all conjunctions are of **one** pair only, the automaton is a "regular" PDA.

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

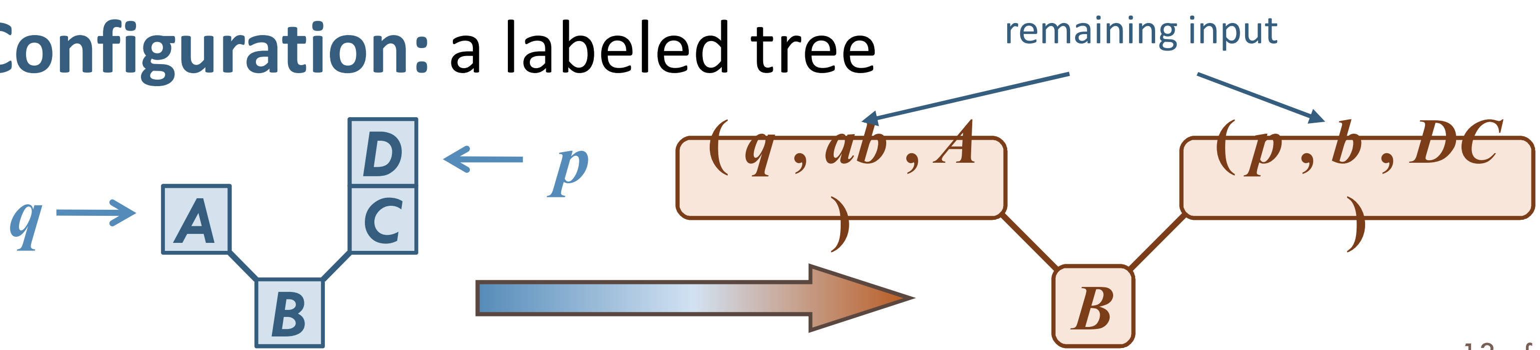
$$A = (\ Q\ ,\ \Sigma\ ,\ \Gamma\ ,\ q_0\ ,\ \delta\ ,\ \perp\ )$$

states — terminals — stack symbols — initial state — transition function — initial stack symbol

- **Transition function:**

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

- **Configuration:** a labeled tree



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

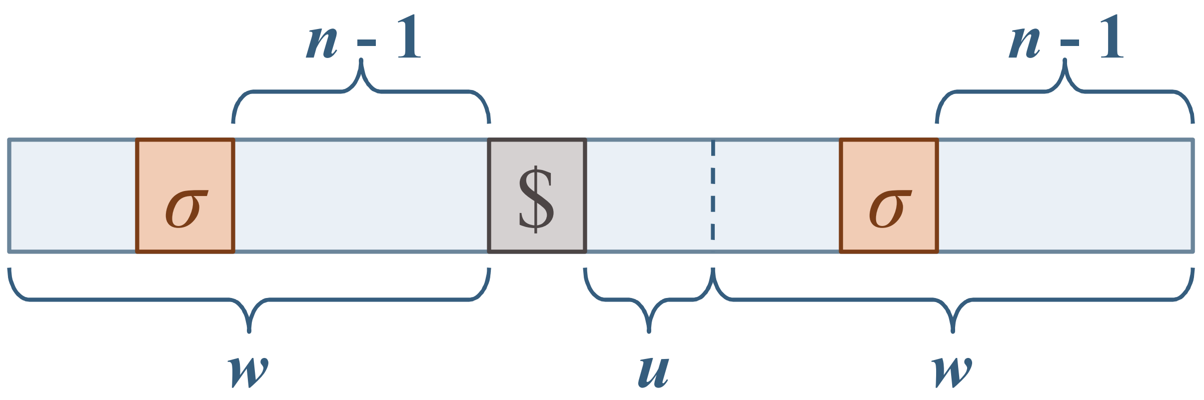
- **Initial Configuration:** $\boxed{\perp} \leftarrow q_0$ $(q_0, w, \perp)$
- **Accepting configuration:** $\boxed{\varepsilon} \leftarrow q$ $(q, \varepsilon, \varepsilon)$
- **Language:** $L(A) = \{w \in \Sigma^* \mid \exists q \in Q, (q_0, w, \perp) \vdash^* (q, \varepsilon, \varepsilon)\}$
- **Note:** all branches must empty ~ 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.



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

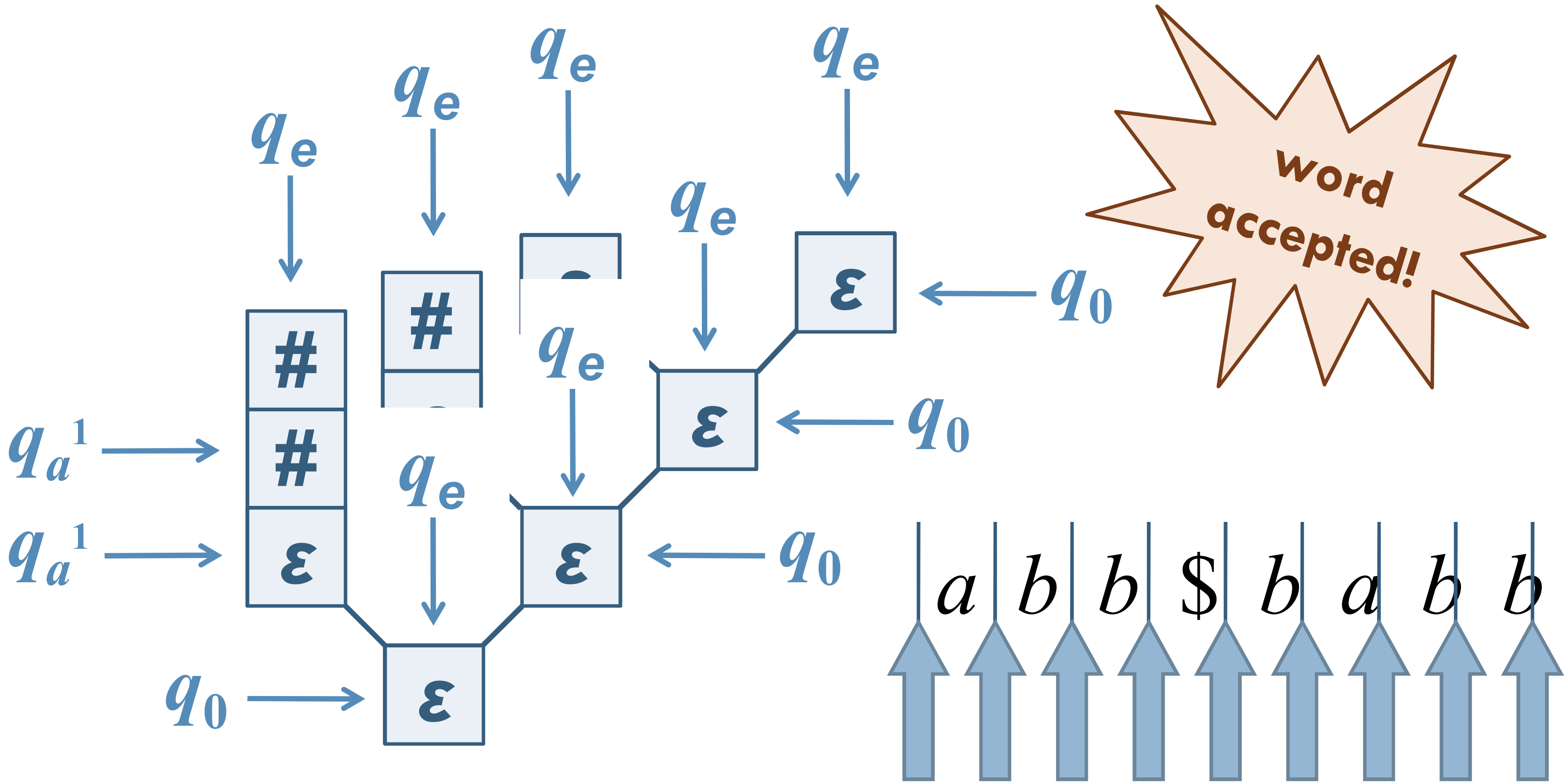
$\delta(q_\sigma{}^2,\tau\neq\sigma,X)=\{(q_\sigma{}^2,X)\}$ — look for σ

$\delta(q_\sigma{}^2,\sigma,X)=\{(q_e,X),(q_\sigma{}^2,X)\}$ — "guess" that this is the σ we are looking for, or keep looking

$\delta(q_e,\tau,\#)=\{(q_e,\varepsilon)\}$ — "count" symbols from σ to the end

$\delta(q_e,\varepsilon,\perp)=\{(q_e,\varepsilon)\}$ — all done: empty stack

Computation of SAPDA for RCM



Main Results

Equivalence Results

Linear CG and One-turn SAPDA

Equivalence Results

- **Theorem 1.** *A language is generated by an CG if and only if it is accepted by an SAPDA.*

- The equivalence is very similar to the classical equivalence between CFG and PDA.

- The proofs of the equivalence are extended versions of the classical proofs.

“only if” Proof Sketch

- Given an CG, we construct an single-state SAPDA using an extension of the classical construction.
- A simulation of the derivation is run in the stack:
  - If the top stack symbol is a non-terminal, it is replaced with the r.h.s. of one of its rules.
  - If the top stack symbol is a terminal, it is emptied while reading the same terminal symbol from the input.
- A correlation is achieved between the stack contents and the grammar sentential forms.

"only if" Proof Simulation

$$S \Rightarrow^* w\,A\,\alpha$$

$$\Downarrow$$

$$w(uB\beta\,\&\cdots)\alpha$$

$$\Downarrow$$

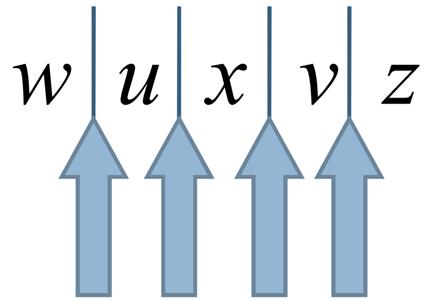
$$\cdots\; ux\,C\gamma\beta \;\cdots$$

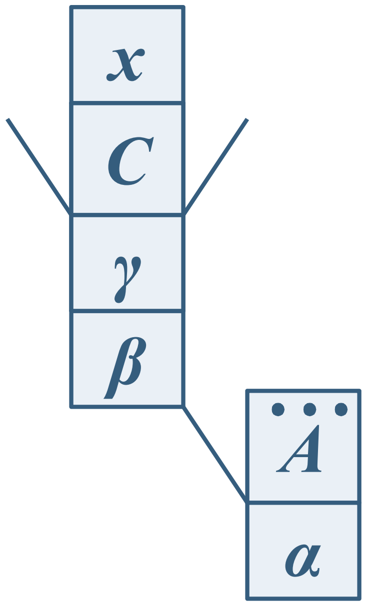
$$\Downarrow_*$$

$$\cdots\; ux(v\&\;\cdots\&v)\gamma\beta \;\cdots$$

$$\Downarrow_*$$

$$\cdots\; ux\;\,v\;\gamma\beta \;\cdots$$





switch variable with terminals

empty rule

“if” Proof Sketch

- Given an SAPDA we construct an CG.
- The proof is an extension of the classical one.
- However, due to the added complexity of the extended models, it is more involved.
- Therefore, we won’t get into it now…

Single-state SAPDA

- The “if” proof translates a general SAPDA into a Conjunctive Grammar.
- The “only if” proof translates a Conjunctive Grammar into a **single-state** SAPDA.

- **Corollary:** Single-state SAPDA and multi-state SAPDA are equivalent.

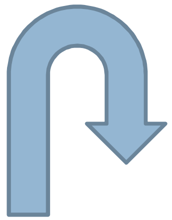
- This characterizes classical PDA as well.

Linear CG and One-turn SAPDA

- Linear Conjunctive Grammars (LCG) [Okhotin, 2001] are an interesting sub-class of CG.
  - Have especially efficient parsing [Okhotin, 2003]
  - Equivalent to Trellis Automata [Okhotin, 2004]
- A conjunctive grammar is **linear** if all conjuncts in all rules contain at most one variable.
- We define a sub-class of SAPDA, **one-turn SAPDA**, and prove equivalence to LCG.

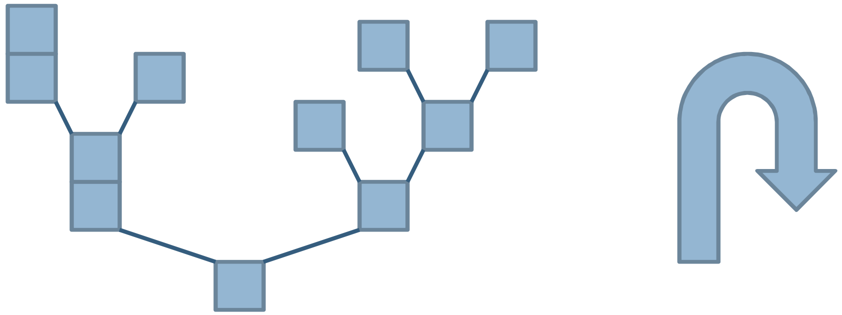
Motivation

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



One-turn SAPDA

- We introduce a sub-family of SAPDA, **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.

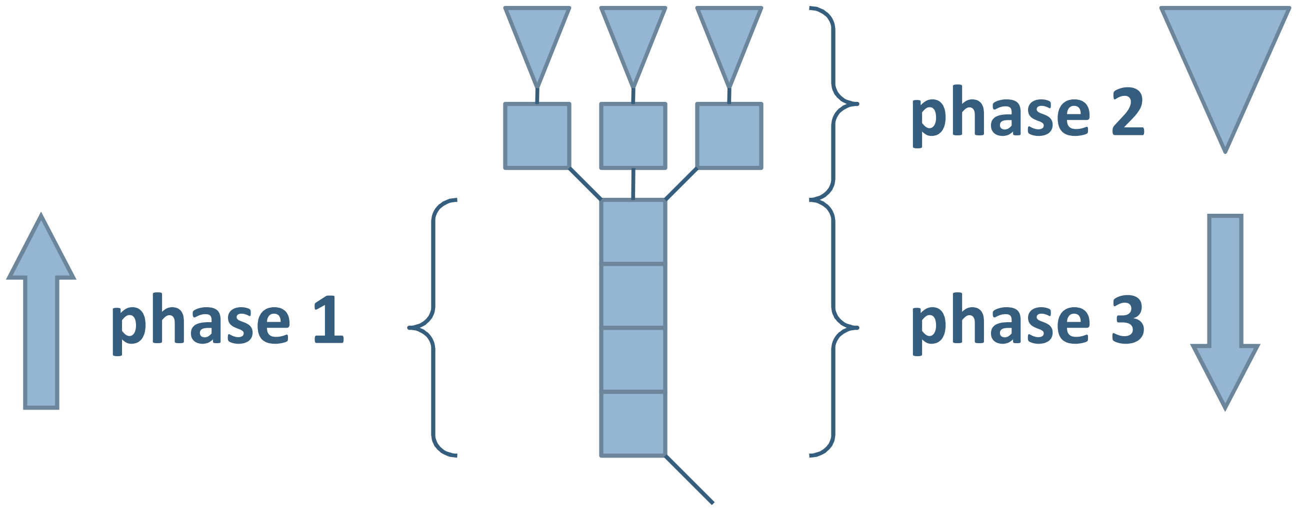


- **Note:** the requirement of a turn is not limiting as we are considering acceptance by empty stack.

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.

Equivalence Results

- **Theorem 2.** *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.

Conjunctive Languages

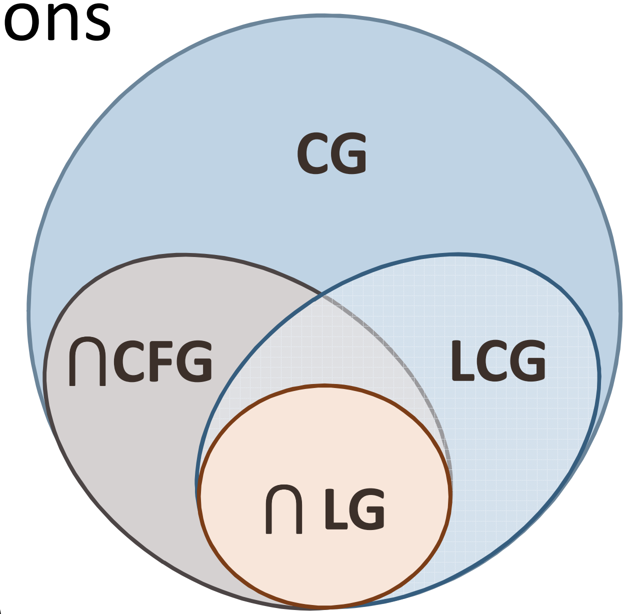
Characterization of Language Class

A Toy Programming Language

Mildly Context Sensitive Languages

Generative Power

- CG can derive all finite conjunctions of CF languages as well as some additional languages (*e.g.*, RCM).
- Linear CG can derive all finite conjunctions of linearCF languages as well as some additional languages (*e.g.*, RCM).
- However, there are some CF languages that cannot be derived by a Linear CG.



Closure Properties

- Union, concatenation, intersection, Kleene star ✓
  - Proven quite easily using grammars
- Homomorphism ✗
- Inverse homomorphism ✓
  - We'll touch on the proof of this in the next slide…

- Linear CL are closed under complement. ✓
- It is an open question whether general CG are closed under complement. ?

Inverse Homomorphism

Model	Technique	Length	Linear CG
CG	Non-classical and complicated	13 pages	Requires separate proof

For the grammar based proof, see [Okhotin, 2003].

Decidability Problems

- **Linear CG Membership:**

  $O(n^2)$ time and $O(n)$ space. ✓

- **General CG Membership:**

  $O(n^3)$ time and $O(n^2)$ space. ✓

- Emptiness, finiteness, equivalence, inclusion, regularity ✗

A Toy Programming Language

- A program in **`PrintVars`** has three parts:
  - Definition of variables
  - Assignment of values
  - Printing of variable values to the screen
- **Example:**

```
VARS  a , b , c
VALS  b = 2 , a = 1 , c = 3
PRNT  b a a c b
```

- **Output:** **2 1 1 3 2**

`PrintVars` Specification

- A **`PrintVars`** program is well-formed if:
  - **(1)** It has the correct structure
  - **(2)** All used variables are defined
  - **(3)** All defined variables are used
  - **(4)** All defined variables are assigned a value
  - **(5)** All variables assigned a value are defined
- Item **(1)** is easily defined by a CF Grammar.
- However, items **(2)** – **(5)** amount to a language reducible to RCM, which is not CF.

A (partial) CG for `PrintVars`

- $S \rightarrow ($ *structure* & *defined_used* & *used_defined* & *defined_assigned* & *assigned_defined* $)$
- *structure* $\rightarrow$ *vars vals prnt*
- *vars* $\rightarrow$ `VARS...` ; *vals* $\rightarrow$ `VALS...` ; *prnt* $\rightarrow$ `PRNT...`
- *defined_used* $\rightarrow$ `VARS` *check_du*
- *check_du* $\rightarrow$ (`a` *X vals X* `a` *X* & `a` *check_du*) | (`b` *X vals X* `b` *X* & `b` *check_du*) | $\cdots$ | *vals X*)
- $X \rightarrow$ `a` $X \mid \cdots \mid$ `z` $X \mid$ `0` $X \mid \cdots \mid$ `9` $X \mid$ `=` $X \mid \varepsilon$

Mildly Context Sensitive Languages

- Computational Linguistics pursues a computational model which **exactly describes** natural languages.
- Originally, context-free models were considered.
- However, non-CF natural language structures led to interest in a slightly extended class of languages – Mildly Context-sensitive Languages (MSCL).
- Several formalisms (*e.g.*, Tree Adjoining Grammars) are known to converge to MCSL. [Vijay-Shanker, 1994]

Conjunctive Languages and MCSL

- We explore the correlation between Conjunctive Languages and MCSL.
- MCSL are loosely categorized as follows:
  (1) They contain the context-free languages ✓
  (2) They contain multiple-agreement, cross-agreement and reduplication ✓
  (3) They are polynomially parsable ✓
  (4) They are semi-linear ✗

- $\Rightarrow$ Not an exact characterization of natural languages, but still with **applicative potential**.

Concluding Remarks

- Summary
- Future Directions

Summary

- Conjunctive Languages are an interesting language class because:
  - They are a strong, rich class of languages.
  - They are polynomially parsable.
  - Their models of computation are intuitive and easy to understand; highly resemble classical CFG and PDA.
- SAPDA are the first automaton model presented for Conjunctive Languages.
- They are an natural extension of PDA.
- They lend new intuition on Conjunctive Languages.

Future Directions

- Broadening the theory of SAPDA
  - Deterministic SAPDA
  - Possible implications on LR-Conjunctive Grammars

- Considering possible applications
  - Formal verification
  - …

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.** *Formal Grammars: Bordeaux 2009.* LNAI 5591. *To appear.*
- 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.: **Conjunctive languages are closed under inverse homomorphism**. *Technical Report 2003-468.* School of Computing, Queens Univ., Kingston, Ontario, Canada.
- Okhotin, A.: **On the equivalence of linear conjunctive grammars and trellis automata.** *RAIRO Theoretical Informatics and Applications.* 38(1) (2004) 69 – 88
- Vijay-Shanker, K., Weir, D.J.: **The equivalence of four extensions of context-free grammars.** *Mathematical Systems Theory*. 27(6) (1994) 511 – 546.