5’-3’ WK FINITE and PUSHDOWN AUTOMATA

Benedek Nagy

Eastern Mediterranean University, Famagusta

nbenedek.inf@gmail.com

Brno, 2017

Outline of the talk

- Chomsky hierarchy (preliminaries)
  - 5’-3’ WK finite automata
  - Pushdown automata
- 5’-3’ WK pushdown automata
  - Definition
  - Examples
  - Some results (semi-linearity, pumping property)
- Concluding remarks

An extended Chomsky hierarchy

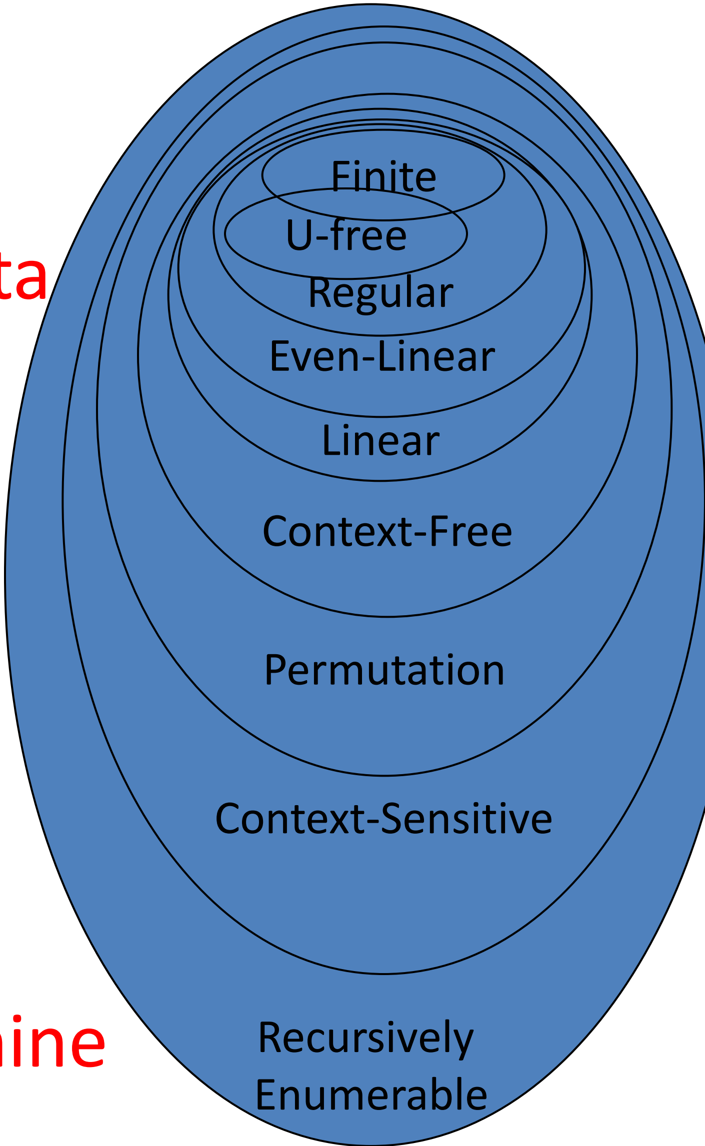
- Finite
- Union-free regular
- Regular
- Even/Fix-rated linear
- Linear
- Context-free
- Permutation
- Context-sensitive
- Recursively enumerable

Finite automata

5'-3' WK finite automata

Pushdown automata

LBA,
Turing machine



Automata for the Chomsky hierarchy

Finite automata

- Regular

2-head finite automata

- Linear

Pushdown automata

- Context-free
-

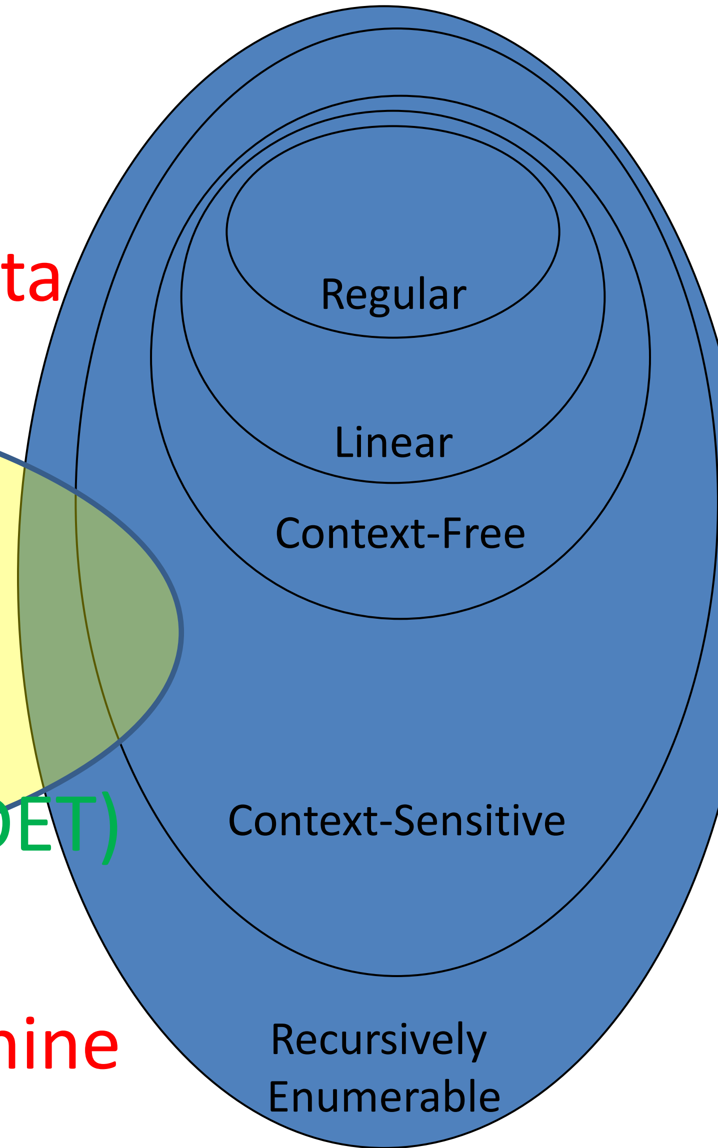
NEW CLASSES (DET-NonDET)

- Context-sensitive

LBA,

- Recursively enumerable

Turing machine



Motivation

- Context-free grammars/languages are popular
  - Theory well-developed
  - Several applications
- Non context-free
  - In several cases, CF is not enough
  - CS is to large (very complex languages are included)
- AIM: larger than CF, but moderate complexity
- Regular-linear
  (finite automata – 2-head finite automata)
  analogy

Finite automata

- (Q,s,V,F,d)
  Q: set of states, s: initial state (in Q)
  V: input alphabet (terminal alphabet in grammars)
  F: set of final states (subset of Q)
  d: transition function
- Deterministic: $d:QxV\rightarrow Q$
- Non-deterministic: $d:Qx(VU\{\lambda\})\rightarrow 2^Q$
  (ε is also used in the role of the empty word)

Linear languages

- Definition by grammar:

$$A \to v,\ A \to vBw$$

- Normal form for the grammar:

$$A \to aB, A \to Ba, A \to a\ (A, B \in N, a \in T)$$

- Even-linear languages (normal form):

$$A \to aBb, A \to a. A \to \lambda$$

Linear languages – parallelism in automata

- Finite automata
  - With 2 heads: reading the word from the beginning and from its end, parallely:

  (Q,s,V,d,F) non-deterministic version:

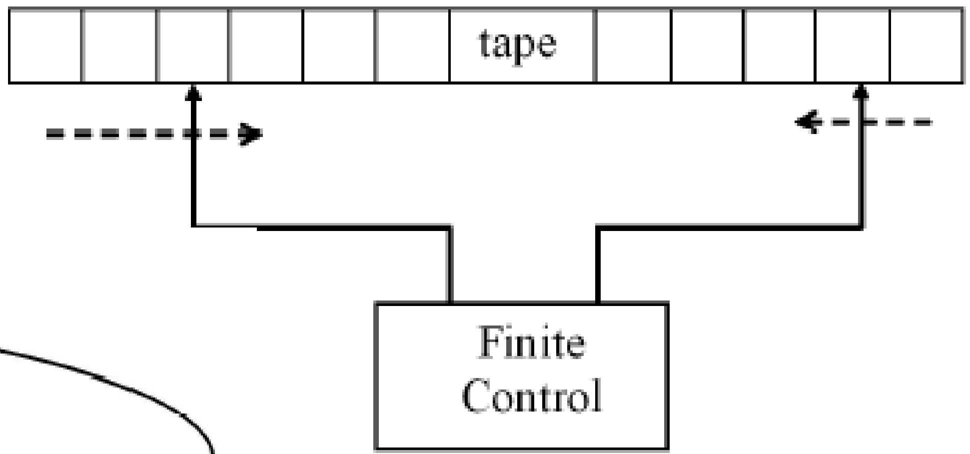
  $d:Qx(VU\{\lambda\})x(VU\{\lambda\})\to 2^{Q}$

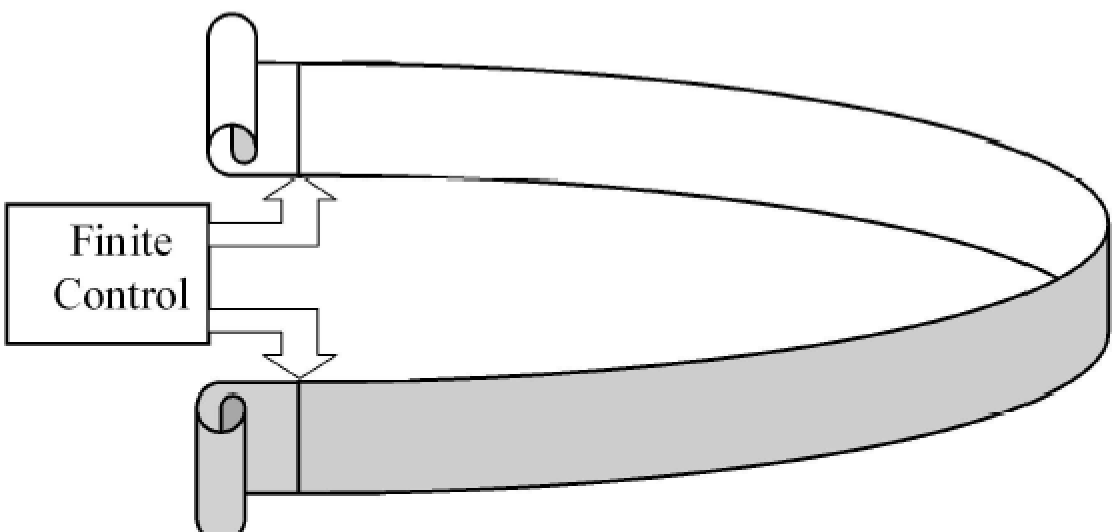
  (deterministic version, if at most 1 transition allowed in any configuration, i.e., QxV*)

Linear languages – 2-head finite automata

$$\langle Q, s, V, d, F \rangle$$

$$d : Q \times (V \cup \{\varepsilon\}) \times (V \cup \{\varepsilon\}) \to 2^Q$$

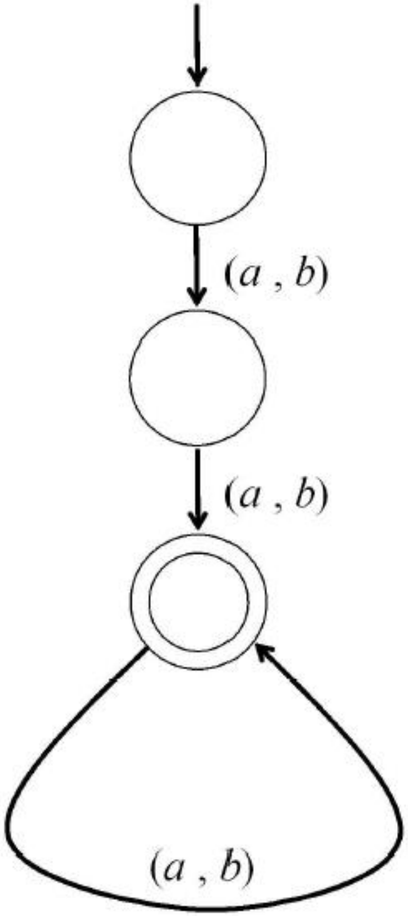




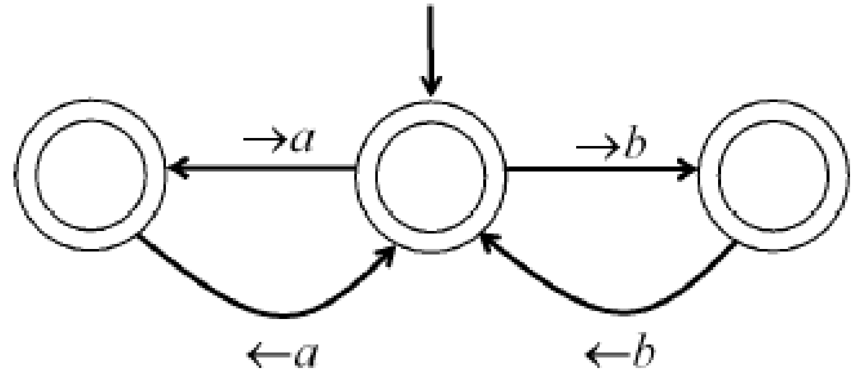
2-head automata - results

- the non-deterministic 2-head automata accept exactly the linear languages.
- for each 2-head automaton there is an equivalent one (with 2-head), in which in each step (transaction) only one of the heads are moving. (normal form)
- The deterministic version of the 2-head automata is weaker : new class: 2detLin

Examples (two possible notions)



- Palindromes:



- $a^n b^n \ (n > 1)$

Pushdown automata

(input) tape, finite control, stack (memory)

(nondeterministic) :

- Σ tapealphabet, Q set of states, q_0 initial state
- Γ stackalphabet, Z_0 initial symbol in the stack
- δ transition function: $(T \cup \{\lambda\}) x Q x \Gamma) \rightarrow 2^{\Gamma^* x Q}$ (finite)

Configuration: (v,q,z)

- v the remaining part of the input word
- z the contents of the stack; q actual state
- initial: (w, q_0, Z_0), accepting: (λ, q, λ) OR (λ, q_f, z)

2-Head Pushdown Automata (2hpda)

- The ordered septuple $M = (Q,\Sigma,\Gamma,\delta\ s,Z,F)$ is a 2-head pushdown automaton (2hpda), where
  - Q is the finite set of states,
  - $s \in Q$ is the initial state,
  - $F \subseteq Q$ is the set of final (or accepting) states,
  - Σ,Γ are the input and stack alphabets with
  - the initial stack symbol $Z \in \Gamma$.
  - The transition function δ is defined as a mapping from $Qx(\Sigma\cup\{\lambda\})^2x\Gamma$ into finite subsets of $Q \times \Gamma^*$.

How 2hpda works

A *configuration* of a 2hpda is a triplet (q, v, y) containing the actual state q, the unread (unprocessed) part v of the input and the actual content y of the stack (the top element is written as the first symbol of y.) The initial configuration of M on input w is (s, w, Z). The transitions of M are defined between pairs of its configurations: $(q, avb, Xy) \vdash (q', v, xy)$, where $q, q' \in Q$, $a, b \in \Sigma \cup \{\lambda\}$, $v \in \Sigma^*$, $y, x \in \Gamma^*$ and $(q', x) \in \delta(q, a, b, X)$. The reflexive and transitive closure of this relation is denoted by $\vdash^*$ (as usual).

M accepts the input $w \in \Sigma^*$ *by a final state* if $(s, w, Z) \vdash^* (q, \lambda, y)$ with a $q \in F$ $(y \in \Gamma^*)$. The *language* $L_f(M)$ accepted by M by final state contains exactly those words that M accepts by a final state. Further, let $\mathcal{L}_f$ denote the family of languages that are accepted by some 2hpda by final state.

The input $w \in \Sigma^*$ is *accepted* by M *by empty stack* if $(s, w, Z) \vdash^* (q, \lambda, \lambda)$ with any $q \in Q$. Consequently, the set of words accepted in this way form the *language* $L_e(M)$ accepted (or recognized) by M by empty stack. The class of languages for that there are some 2hpda that accept them by empty stack is denoted by $\mathcal{L}_e$.

First results

- The class of languages that are accepted by empty-stack by some 2hpda and the class of languages that are accepted by nal state by some 2hpda are the same.
- Notation: $\mathcal{L}_{2hpda}$ instead of $\mathcal{L}_f$ and $\mathcal{L}_e$
- *The class* $\mathcal{L}_{2hpda}$ *contains all context-free languages.*

Technical results – normal forms

1. A 2hpda is in head normal form if in each transition at most one of its heads moves.
2. A 2hpda is in stack normal form if each of its transitions is

- *either a clear pop, i.e., it is of type* $(p, \lambda) \in \delta(q, a, b, X)$ *(*$p, q \in Q$*,* $a, b \in \Sigma \cup \{\lambda\}$*,* $X \in \Gamma$*);*
- *or a push with exactly one stack symbol, i.e.,* $(p, YX) \in \delta(q, a, b, X)$ *(*$p, q \in Q$*,* $a, b \in \Sigma \cup \{\lambda\}$*,* $X, Y \in \Gamma$*);*
- *or the stack does not change:* $(p, X) \in \delta(q, a, b, X)$ *(*$p, q \in Q$*,* $a, b \in \Sigma \cup \{\lambda\}$*,* $X \in \Gamma$*).*

3. A 2hpda is in strong normal form if it is in both head normal form and stack normal form.

- Now, to underline the efficiency of 2hpda’s some interesting examples are presented.
- But, first, we recall the concept of Mildly context-sensitive language classes

Mildly context-sensitive languages

- From motivation of formal linguistics
- Mildly context-sensitive classes of languages
  - Containing all CF languages
  - Containing only semi-linear languages
  - Polynomial word problem
  - They contain the 3 linguistically important non context-free languages:

Mildly CS examples

Example 4.1 *Let* $M = (\{s,p,q\},\{a,b,c\},\{Z,X\},\delta,s,Z,\{q\})$ *be a* determinsitic *2hpda, where* δ *is defined as follows:*

$(s, XZ) \in \delta(s, a, c, Z) \quad (s, XX) \in \delta(s, a, c, X) \quad (p, \lambda) \in \delta(s, b, \lambda, X)$

$(p, \lambda) \in \delta(p, b, \lambda, X) \quad (q, \lambda) \in \delta(p, \lambda, \lambda, Z).$

- aaabbbccc s Z
- aabbbcc s XZ
- abbbc s XXZ
- bbb s XXXZ
- bb p XXZ
- b p XZ
- \- p Z
- \- q - ACCEPT

Mildly CS examples

Example 4.1 *Let* $M = (\{s,p,q\},\{a,b,c\},\{Z,X\},\delta,s,Z,\{q\})$ *be a 2hpda, where* δ *is defined as follows:*

$(s,XZ) \in \delta(s,a,c,Z) \quad (s,XX) \in \delta(s,a,c,X) \quad (p,\lambda) \in \delta(s,b,\lambda,X)$

$(p,\lambda) \in \delta(p,b,\lambda,X) \qquad (q,\lambda) \in \delta(p,\lambda,\lambda,Z).$

- The language of triple (multiple) agreement

$$L_{abc} = \{a^n b^n c^n \mid n > 0\}$$

is accepted.

Mildly CS examples

determinsitic

Example 4.2 *Let* $M = (\{s,p,q,r\},\{a,b,c\},\{Z,X\},\delta,s,Z,\{r\})$ *be a 2hpda, where* δ *is defined as follows:*

$(s,XZ) \in \delta(s,a,\lambda,Z)$ $(s,XX) \in \delta(s,a,\lambda,X)$ $(p,X) \in \delta(s,b,d,X)$

$(p,X) \in \delta(p,b,d,X)$ $(q,\lambda) \in \delta(p,c,\lambda,X)$ $(q,\lambda) \in \delta(q,c,\lambda,X)$

$(r,\lambda) \in \delta(q,\lambda,\lambda,Z)$.

- The language of crossed dependencies

$$L_{abcd} = \{a^n b^m c^n d^m \mid n,m > 0\}$$

is recognized.

Mildly CS examples

Example 4.3 *Let* $M = (\{s,p,q\},\{a,b\},\{Z,A,B\},\delta,s,Z,\{q\})$ *be a 2hpda, where* δ *is defined as follows:*

$(s,AZ) \in \delta(s,a,\lambda,Z)$ $\quad (s,BZ) \in \delta(s,b,\lambda,Z)$ $\quad (s,AA) \in \delta(s,a,\lambda,A)$

$(s,BA) \in \delta(s,b,\lambda,A)$ $\quad (s,AB) \in \delta(s,a,\lambda,B)$ $\quad (s,BB) \in \delta(s,b,\lambda,B)$

$(p,A) \in \delta(s,\lambda,\lambda,A)$ $\quad (p,B) \in \delta(s,\lambda,\lambda,B)$ $\quad (p,\lambda) \in \delta(p,\lambda,a,A)$

$(p,\lambda) \in \delta(p,\lambda,b,B)$ $\quad (q,\lambda) \in \delta(p,\lambda,\lambda,Z).$

- The accepted language is the copy language, that is

$$L_{ww} = \{ww \mid w \in \{a,b\}^+\}.$$

- Observe: non-deterministic.

Mildly CS examples

determinsitic

Example 4.4 *Let* $M = (\{s, p, q\}, \{a, b, c\}, \{Z, A, B\}, \delta, s, Z, \{q\})$ *be a 2hpda, where* δ *is defined as follows:*

$(s, AZ) \in \delta(s, a, \lambda, Z)$ $(s, BZ) \in \delta(s, b, \lambda, Z)$ $(s, AA) \in \delta(s, a, \lambda, A)$

$(s, BA) \in \delta(s, b, \lambda, A)$ $(s, AB) \in \delta(s, a, \lambda, B)$ $(s, BB) \in \delta(s, b, \lambda, B)$

$(p, A) \in \delta(s, c, \lambda, A)$ $(p, B) \in \delta(s, c, \lambda, B)$ $(p, \lambda) \in \delta(p, \lambda, a, A)$

$(p, \lambda) \in \delta(p, \lambda, b, B)$ $(q, \lambda) \in \delta(p, \lambda, \lambda, Z).$

- Marked copy is accepted:

$$L_{wcw} = \{wcw \mid w \in \{a, b\}^+\}.$$

Properties of 2HPDA Languages

- Each 2hpda language is semi-linear.

The Parikh image of a string w over an ordered alphabet $\{a_1, \ldots, a_k\}$ is the vector $\Psi(w) = (m_1, \ldots, m_k)$ of non-negative integers such that m_i is the number of occurrences of a_i in w. The Parikh image of a language L is the set of vectors $\Psi(L) = \{\Psi(w) \mid w \in L\}$. Two languages are *letter equivalent* if their Parikh images coincide.

A set of the form $\{\alpha_0 + n_1\alpha_1 + \cdots + n_m\alpha_m \mid n_j \geq 0 \text{ for } j = 1, 2, \ldots, m\}$, where $\alpha_0, \alpha_1, \ldots, \alpha_m$ are vectors of non-negative integers, is said to be a *linear* set. A *semi-linear* set is a finite union of linear sets. A language is semi-linear if its Parikh image is semi-linear. It is well-known [17] that the Parikh images of regular languages and Parikh images of context-free languages coincide with semi-linear sets; but there are non semi-linear context-sensitive languages.

Properties of 2HPDA Languages

- Each 2hpda language is semi-linear.

The Parikh image of a string w over an ordered alphabet $\{a_1, \ldots, a_k\}$ is the vector $\Psi(w) = (m_1, \ldots, m_k)$ of non-negative integers such that m_i is the number of occurrences of a_i in w. The Parikh image of a language L is the set of vectors $\Psi(L) = \{\Psi(w) \mid w \in L\}$. Two languages are *letter equivalent* if their Parikh images coincide.

A set of the form $\{\alpha_0 + n_1\alpha_1 + \cdots + n_m\alpha_m \mid n_j \geq 0 \text{ for } j = 1, 2, \ldots, m\}$, where $\alpha_0, \alpha_1, \ldots, \alpha_m$ are vectors of non-negative integers, is said to be a *linear* set. A *semi-linear* set is a finite union of linear sets. A language is semi-linear if its Parikh image is semi-linear. It is well-known [17] that the Parikh images of regular languages and Parikh images of context-free languages coincide with semi-linear sets; but there are non semi-linear context-sensitive languages.

- HINT: 2hpda → letter equivalent PDA

The place in the hierarchy

The class is $\mathcal{L}_{2hpda}$ *strictly between the context-free and context-sensitive classes.*

Closure properties

The language class accepted by the class of 2-head pushdown automata is closed under operations

- union,
- reversal and
- homomorphisms.

NON-Closure properties

We need some tools…

Pumping of 2hpda languages

- Let L be a 2hpda language. If L is infinite, then there is a value $n \in N$ such that each word $w \in L$ with $|w| > n$ can be written in the form $w = u_0v_1u_1v_2u_2v_3u_3v_4u_4$ such that $|v_1v_2v_3v_4| > 0$ and for each $r \in N$ $u_0v_1^ru_1v_2^ru_2v_3^ru_3v_4^ru_4 \in L.$

CF-composability

- Let *L* be a 2hpda language. Then,

there exist two context-free languages L_1 *and* L_2 *with the* following properties

- *every* $w \in L$ *can be factorized to* $w = uv$*, such that* $u \in L_1$*,* $v \in L_2$*;*
- *for every word* $u \in L_1$ *there is a word* $v \in L_2$ *such that* $uv \in L$*; and*
- *for every word* $v \in L_2$ *there is a word* $u \in L_1$ *such that* $uv \in L$*.*

NON-Closure properties

The language class accepted by the class of 2-head pushdown automata is NOT closed under operations

- intersection,
- complement (union-yes,intersection-no→NO)
- concatenation
- and square, Kleene-star, Kleene-plus.

Some relations

New type of automata:

Finite Automata → Pushdown Automata

2-head finite automata → 2hpda

Some relations

New type of automata:

Finite Automata	Pushdown Automata
Regular	CF
2-head finite automata	2hpda
Linear	NEW CLASS of LANG.

NEW CLASS of LANG.

Semi-linear, all CF lang. are included

Important mildly CS languages

Pumping and closure properties

Parsing (in P, actually, n^5)

Special subclasses of 5'-3' WK-PDA:
deterministic, stateless,

- to introduce the determinsistic variants, acceptance with final states will be more important

- We have seen some examples already: marked copy, multiple agreement, cross dependencies.

2hPDA and control PDA

Definition 3.1 *Let $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$ be a 2hPDA in head normal form. Let $\Sigma' := \{\overleftarrow{a} \mid a \in \Sigma\} \cup \{\overrightarrow{a} \mid a \in \Sigma\}$, and let $M' = (Q, \Sigma', \Gamma, q_0, \bot, F, \delta')$ be a (1h)PDA, where we define δ' as follows:*

- *let $(q', s') \in \delta'(q, \overleftarrow{a}, s)$, if and only if, $(q', s') \in \delta(q, a, \lambda, s)$, where $q, q' \in Q$, $s \in \Gamma$, $s' \in \Gamma^*$ and $a \in \Sigma$;*
- *let $(q', s') \in \delta'(q, \overrightarrow{a}, s)$, if and only if, $(q', s') \in \delta(q, \lambda, a, s)$, where $q, q' \in Q$, $s \in \Gamma$, $s' \in \Gamma^*$ and $a \in \Sigma$;*
- *let $(q', s') \in \delta'(q, \lambda, s)$, if and only if, $(q', s') \in \delta(q, \lambda, \lambda, s)$, where $q, q' \in Q$, $s \in \Gamma$ and $s' \in \Gamma^*$.*

We call M' the control PDA of M.

Theorem 3.2 *Let $M = (Q, \Sigma, \Gamma, q_0, \bot, F, \delta)$ be a 2hPDA in head normal form, and let $M' = (Q, \Sigma', \Gamma, q_0, \bot, F, \delta')$ be its control PDA. Then M is deterministic, if and only if, both (i) and (ii) hold.*

(i) M' is deterministic, and

(ii) for every $q \in Q$ and $r \in \Gamma$, at most one of the following can be true:

(ii/a) $\exists a \in \Sigma : \delta'(q, \overleftarrow{a}, r) \neq \emptyset$,

(ii/b) $\exists a \in \Sigma : \delta'(q, \overrightarrow{a}, r) \neq \emptyset$,

(ii/c) $\delta'(q, \lambda, r) \neq \emptyset$.

Properties of det2HPDA

- A deterministic PDA may run into loops during input processing, if λ-movements are allowed. The PDA may stuck in an infinite loop of λ -movements either leaving unprocessed letters on the input tape, or it may successfully read the input word, and then get into an infinite loop. These loops can be eliminated.
- We have proven similar result for det2hPDA:
- Each det2hpda is equivalent with a loop-free det2hpda.

Properties of det2HPDA Languages

- The deterministic 2hPDA language family contains the deterministic context-free language family and also det.LIN and 2detLIN families.
- Based on loop elimination, the deterministic 2hPDA language family is closed under complement.
- This class is incomparable with LIN and CF.
- Anti-closure properties: it is not closed under union, intersection, concatenation, Kleene-star.
- Closure: it is closed under reversal (detCF is NOT!); closed under intersection with regular languages

Stateless / simple variants

- Every language of the class of 5’ – 3’ WK pda languages can be accepted by a stateless (N) 5’ – 3’ WK pda.
- Every language of the class of 5’ – 3’ WK pda languages can be accepted by a 5’ – 3’ WK pda with the property that at most one of the heads read some symbol(s) in each transition.

Thank you!

1. B. Nagy: On 5'→3' sensing Watson-Crick finite automata (2007), DNA13, The 13th International Meeting on DNA Computing, Memphis, Tennessee, USA, 327-336. (preproceedings)
2. B. Nagy: On 5'→3' sensing Watson-Crick finite automata, DNA 13, Lecture Notes in Computer Science - LNCS 4848 (2008), 256-262.
3. On a hierarchy of 5'→3' sensing WK finite automata languages, CiE 2009, Computability in Europe 2009: Mathematical Theory and Computational Practice, Abstract Booklet, University of Heidelberg, Germany, 266-275.
4. with. P. Leupold, 5'→3' Watson-Crick automata with several runs, Workshop on Non-Classical Models of Automata and Applications (NCMA), (satellite event of the International Symposium on Fundamentals of Computation Theory - FCT), Wroclaw, Poland, (2009) 167-180.
5. 5'→3' Watson-Crick automata with several runs, Fundamenta Informaticae 104 (2010) 71-91. (co-author: Peter Leupold)
6. Benedek Nagy: 5' → 3' Sensing Watson-Crick Finite Automata, Sequence and Genome Analysis II – Methods and Applications, iConcept Press (2010), 39-56.
7. with O. Egecioglu and L. Hegedüs: Stateless Multicounter 5'→3' Watson-Crick Automata, BIC-TA 2010, Fifth IEEE International Conference on Bio-Inspired Computing: Theories and Applications, Liverpool, UK, (Volume II), 1599-1606.
8. Hierarchy Results On Stateless Multicounter 5' → 3' Watson-Crick Automata, IWANN 2011, LNCS 6691 (2011), 465-472. (co-authors: Ö. Egecioglu, L. Hegedüs)
9. Hierarchies of Stateless Multicounter 5' → 3' Watson-Crick Automata Languages, Fundamenta Informaticae - FI 110 (2011), 111-123. (co-authors: Ö. Egecioglu, L. Hegedüs)
10. Benedek Nagy: A class of 2-head finite automata for linear languages, Triangle 8 (Languages. Mathematical Approaches) (2012), 89-99.
11. László Hegedüs, Benedek Nagy, Ömer Egecioglu: Stateless Multicounter 5' → 3' Watson-Crick Automata: The Deterministic Case, Natural Computing 11 (2012), 361-368.
12. B. NAGY, On a hierarchy of 5’ → 3’ sensing Watson-Crick finite automata languages. Journal of Logic and Computation 23 (2013), 855-872.
13. Nagy Benedek: DNS számítógépek és formális modelljeik, Typotex, Budapest, 2014. (DNA computing and its formal models, in Hungarian)
14. Benedek Nagy: A Family Of Two-Head Pushdown Automata, NCMA 2015: 7th Workshop on Non Classical Models of Automata and Applications, Porto, Portugal, 177-191.
15. Dávid Angyal, Benedek Nagy: An extension of the LR parsing algorithm for two-head pushdown automata. NCMA 2017: 71-86