DETERMINISTIC TWO-HEAD PUSHDOWN AUTOMATA and AN EXTENSION OF LR PARSING

Benedek Nagy

Eastern Mediterranean University, Famagusta,

nbenedek.inf@gmail.com

Dávid Angyal

University of Debrecen, Debrecen, Hungary

NCMA 2017, Prague

Outline of the talk

- Chomsky hierarchy (preliminaries)
  - 2-head finite automata, Pushdown automata
  - An extension: 2-head pushdown automata
- (Determininstic) 2-head pushdown automata
  - Definition, Examples
  - Characterization by controlled grammars
  - Some results (semi-linearity, pumping property)
- Extending LR parser

An extended Chomsky hierarchy

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

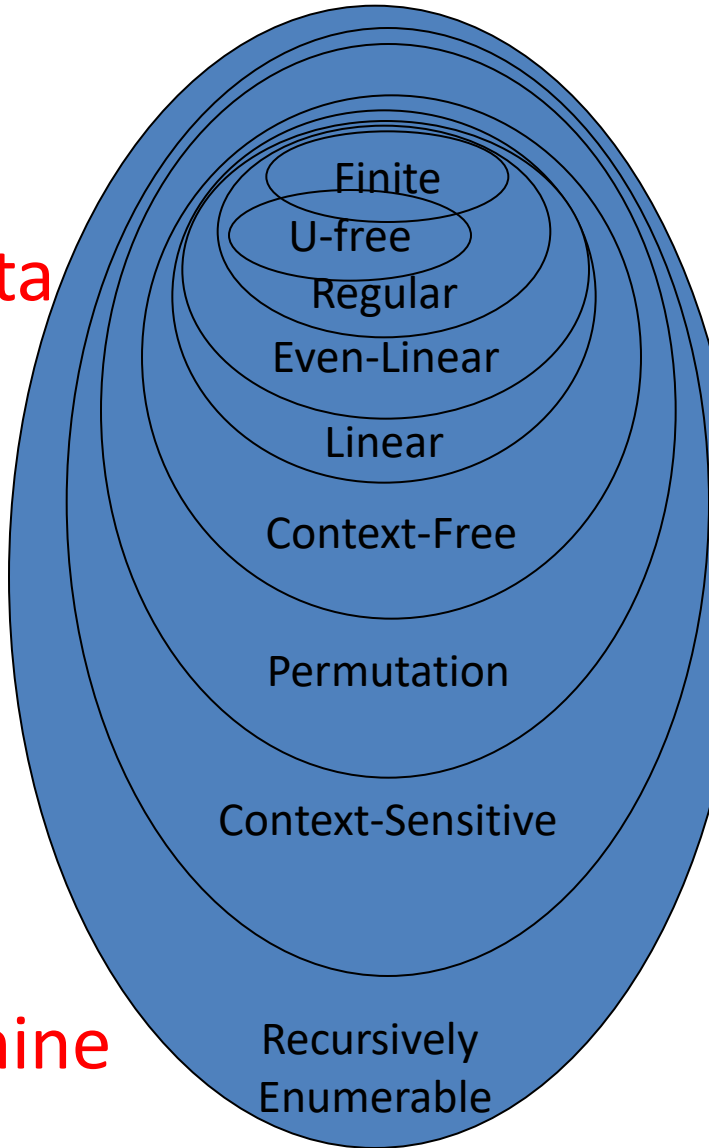
Finite automata

2-head finite automata

Pushdown automata

LBA,

Turing machine



Automata for the Chomsky hierarchy

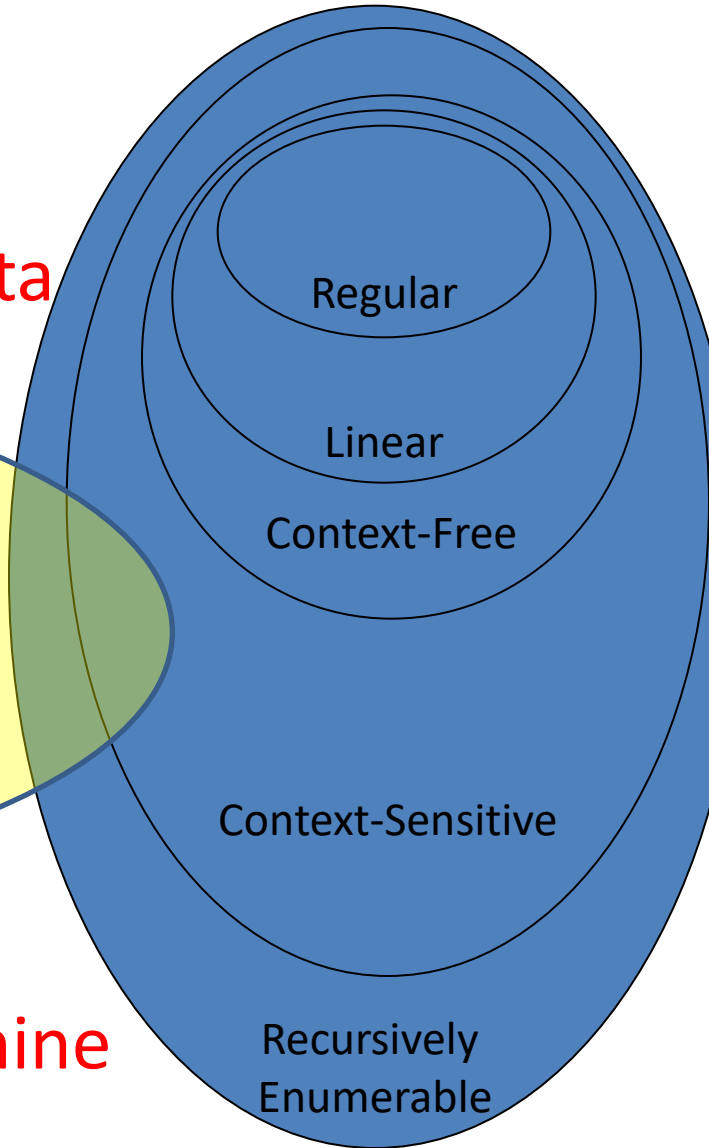
- Regular
- Linear
- Context-free
- Context-sensitive
- Recursively enumerable

Finite automata

2-head finite automata
Pushdown automata

NEW CLASSES: nondet & det.

LBA,
Turing machine



Motivation

- Context-free grammars/languages are popular
  - Theory well-developed
  - Several applications
  - Efficient PARSERS
- Non context-free
  - In several cases, CF is not enough
  - CS is to large (very complex languages are included)
- AIM: go beyond CF, but keep moderate complexity
- Regular-linear
  (finite automata – 2-head finite automata) analogy,
  however, in the latter case det. version is less powerful

Notations // Finite automata

- (Q,q_0,Σ,δ,F)
- Q: set of states, q_0: initial state (in Q)
  Σ : input alphabet (terminal alphabet in grammars)
  F: set of final states (subset of Q)
  δ : transition function
- Deterministic: $\delta : Q \times V \rightarrow Q$
- Non-deterministic: $\delta : Q \times (V \cup \{\lambda\}) \rightarrow 2^Q$
  (ε is also used in the role of the empty word)

Notations // Pushdown automata

$(Q, q_0, \Sigma, \Gamma, Z_0, \delta, F)$

(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: $(T \cup \{\lambda\}) \times Q \times \Gamma) \rightarrow 2^{\Gamma^* \times Q}$ (finite)

Configuration: (v,q,z)

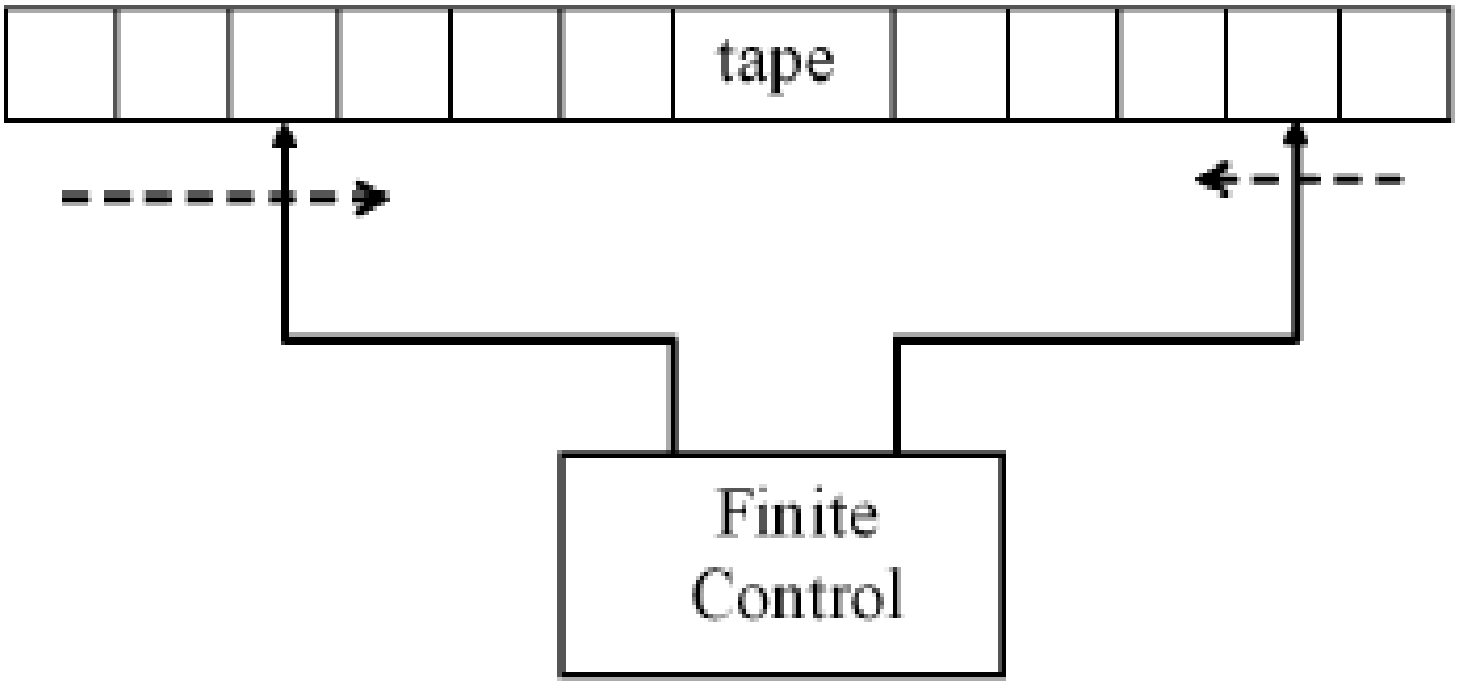
- v the remaining part of the input word
- z the contents of the stack; q actual state
- initial: (w, q_0, Z_0), accepting: (λ, q, λ) OR (λ, q_f, z)

Linear languages

- Definition by grammar: $A \to v,\ A \to vBw$
  Normal form for the grammar:

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

-



2-Head Pushdown Automata (2hpda)

- The ordered septuple $M = (Q,\Sigma,\Gamma,\delta, q_0,Z,F)$ is a 2-head pushdown automaton (2hpda), where
  - Q is the finite set of states,
  - $q_0 \in Q$ 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 $\mathcal{L}_e$.

Normal form & DET2hPDA

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

- A 2hPDA is deterministic, if
  for every configuration (q,w,s),
  there is at most one
  configuration (q',w', s'), such that,
  (q,w,s) $\Rightarrow$ (q',w',s') holds.

Acceptance modes

- The class of languages that are accepted by empty-stack by some 2hpda and the class of languages that are accepted by final 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.*
- NOW, to introduce the determinsistic variants, acceptance with final states will be more important

- Now, to underline the efficiency of determinsitic 2hPDA's, some interesting examples are presented (nonCF, mildly CS languages).

- Let us recall, first, the concept of Mildly context-sensitive language classes

Mildly context-sensitive languages

- From motivation of formal linguistics
- Mildly context-sensitive classes of languages
  - Contain only semi-linear languages
  - Polynomial word problem / efficient parsing
  - They contain the 3 linguistically important non context-free languages:
    - Copy, multiple agreement, cross dependencies
  - (Contain all CF languages) % sometimes omitted

Mildly CS example: multiple agreement

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:*

determinsitic

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

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

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

Mildly CS example: multiple agreement

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:*

determinsitic

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

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

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

determinsitic

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

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

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

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

- Marked copy is accepted:

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

Controlled grammars

- G = (N, T, S, P, L) is a controlled grammar, if
- (N, T, S, P) is a context-free grammar, and
- L is a language over the alphabet P.
- Let $S \Rightarrow_{p_1} w_1 \Rightarrow_{p_2} \ldots \Rightarrow_{p_n} w$
  be a derivation of the word w in (N ,T ,S , P).
- Each word w is in L(G) if $p_1 \ldots p_n \in P$, i.e.,
  $p_1 \ldots p_n$ is in the intersection of L
  and the Szilard-language of (N, T, S, P).

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

LR parsing

- LR parsing algorithms are shift-reduce bottom-up parsers.
- The input is scanned from left to right, and the result is the right-most derivation of the input word.
- LR parsers usually use lookahead symbols, which means in some steps we look ahead what the next few input symbols are without considering those symbols read.

LR parser automaton

- LR parser automaton is constructed from a (CF) grammar
- LR-state is a set of productions with position markers
- From an LR-state q, there is a transition to LR-state q' with the letter X, if q contains an item with the position marker standing before the letter X
- The LR-states are the stack alphabet of the LR automaton.
- The LR automaton uses the topmost stack symbol (LR-state) and the next letter (or next lookahead symbols) to determine whether it has to
  - shift another LR-state onto the stack;
  - reduce the topmost few symbols of the stack (and output the corresponding production);
  - accept or reject the input.
- For every deterministic context-free language, a (deterministic) LR(1) parser can be constructed.

Extension of LR parsing to det2hPDA

- Based on loop-free version and equivalent detPDA controlled grammar.
- Modification: left and right reading:
- Reading a letter $\overleftarrow{a}$ corresponds to reading a letter a from the left-end of the input, &
  reading a letter $\overrightarrow{a}$ corresponds to reading a letter a from the right-end of the input.
- Through on some technical lemmas, it is proven that
  For every deterministic 2hPDA, the LR parsing algorithm is deterministic and has a linear time complexity.

Example

Example 3.10 *Let* $M = (\{q_l, q_r\}, \{a, b, c\}, \{\bot, A, B\}, q_l, \bot, \{q_r\}, \delta)$, *where*

$\delta(q_l, a, \lambda, A) = (q_l, AA)$, $\delta(q_l, a, \lambda, B) = (q_l, BA)$, $\delta(q_l, a, \lambda, \bot) = (q_l, \bot\ A)$

$\delta(q_l, b, \lambda, A) = (q_l, AB)$, $\delta(q_l, b, \lambda, B) = (q_l, BB)$, $\delta(q_l, b, \lambda, \bot) = (q_l, \bot\ B)$

$\delta(q_l, c, \lambda, A) = (q_r, A)$, $\delta(q_l.c, \lambda, B) = (q_r, B)$, $\delta(q_l, c, \lambda, \bot) = (q_r, \bot\ )$

$\delta(q_r, \lambda, a, A) = (q_r, \lambda)$, $\delta(q_r, \lambda, b, B) = (q_r, \lambda)$, $\delta(q_r, \lambda, \lambda, \bot) = (q_r, \lambda)$.

$$M \text{ accepts } \{wcw \mid w \in \{a, b\}^*\} \text{ by empty stack.}$$

1. We can construct an equivalent automaton which accepts the language by final state.
2. We construct the ”grammar”-equivalent.

Example – cont.

- Grammar:

$$S \to (q_l, \bot, q_r)$$
$$(q_l, \bot, q_r) \to \overleftarrow{c}$$
$$(q_l, \bot, q_r) \to \overleftarrow{a}\,(q_l, A, q_r)$$
$$(q_l, \bot, q_r) \to \overleftarrow{b}\,(q_l, B, q_r)$$
$$(q_l, A, q_r) \to \overleftarrow{a}\,(q_l, A, q_r)(q_r, A, q_r)$$
$$(q_l, A, q_r) \to \overleftarrow{b}\,(q_l, B, q_r)(q_r, A, q_r)$$
$$(q_l, A, q_r) \to \overleftarrow{c}\,(q_r, A, q_r)$$
$$(q_l, B, q_r) \to \overleftarrow{a}\,(q_l, A, q_r)(q_r, B, q_r)$$
$$(q_l, B, q_r) \to \overleftarrow{b}\,(q_l, B, q_r)(q_r, B, q_r)$$
$$(q_l, B, q_r) \to \overleftarrow{c}\,(q_r, B, q_r)$$
$$(q_r, A, q_r) \to \overrightarrow{a}$$
$$(q_r, B, q_r) \to \overrightarrow{b}$$

empty prod
FREE

Example – FIRST and FOLLOW sets

- The FIRST set for each terminal is a singleton set containing only the terminal itself. Then,
- the FIRST and the FOLLOW sets for the nonterminals of this grammar are :

nonterminal X	FIRST(X)	FOLLOW(X)
S	$\{\overleftarrow{a}, \overleftarrow{b}, \overleftarrow{c}\}$	$\{\dashv\}$
$(q_l, \perp, q_r)$	$\{\overleftarrow{a}, \overleftarrow{b}, \overleftarrow{c}\}$	$\{\dashv\}$
(q_l, A, q_r)	$\{\overleftarrow{a}, \overleftarrow{b}, \overleftarrow{c}\}$	$\{\dashv, \overrightarrow{a}, \overrightarrow{b}\}$
(q_l, B, q_r)	$\{\overleftarrow{a}, \overleftarrow{b}, \overleftarrow{c}\}$	$\{\dashv, \overrightarrow{a}, \overrightarrow{b}\}$
(q_r, A, q_r)	$\{\overrightarrow{a}\}$	$\{\dashv, \overrightarrow{a}, \overrightarrow{b}\}$
(q_r, B, q_r)	$\{\overrightarrow{b}\}$	$\{\dashv, \overrightarrow{a}, \overrightarrow{b}\}$

Example – LR-states

- $I_0 = \{S \to .(q_l, \bot, q_r), (q_l, \bot, q_r) \to .\overleftarrow{c}, (q_l, \bot, q_r) \to .\overleftarrow{a}(q_l, A, q_r),$
  $(q_l, \bot, q_r) \to .\overleftarrow{b}(q_l, B, q_r)\}$
- $I_1 = \{S \to (q_l, \bot, q_r).\}$
- $I_2 = \{(q_l, \bot, q_r) \to \overleftarrow{a}.(q_l, A, q_r), (q_l, A, q_r) \to .\overleftarrow{a}(q_l, A, q_r)(q_r, A, q_r),$
  $(q_l, A, q_r) \to .\overleftarrow{b}(q_l, B, q_r)(q_r, A, q_r), (q_l, A, q_r) \to .\overleftarrow{c}(q_r, A, q_r)\}$
- $I_3 = \{(q_l, \bot, q_r) \to \overleftarrow{b}.(q_l, B, q_r), (q_l, B, q_r) \to .\overleftarrow{a}(q_l, A, q_r)(q_r, B, q_r),$
  $(q_l, B, q_r) \to .\overleftarrow{b}(q_l, B, q_r)(q_r, B, q_r), (q_l, B, q_r) \to .\overleftarrow{c}(q_r, B, q_r)\}$
- $I_4 = \{(q_l, \bot, q_r) \to \overleftarrow{c}.\}$
- $I_5 = \{(q_l, \bot, q_r) \to \overleftarrow{a}(q_l, A, q_r).\}$
- $I_6 = \{(q_l, A, q_r) \to \overleftarrow{a}.(q_l, A, q_r)(q_r, A, q_r), (q_l, A, q_r) \to .\overleftarrow{a}(q_l, A, q_r)(q_r, A, q_r),$
  $(q_l, A, q_r) \to .\overleftarrow{b}(q_l, B, q_r)(q_r, A, q_r), (q_l, A, q_r) \to .\overleftarrow{c}(q_r, A, q_r)\}$
- $I_7 = \{(q_l, A, q_r) \to \overleftarrow{b}.(q_l, B, q_r)(q_r, A, q_r), (q_l, B, q_r) \to .\overleftarrow{a}(q_l, A, q_r)(q_r, B, q_r),$
  $(q_l, B, q_r) \to .\overleftarrow{b}(q_l, B, q_r)(q_r, B, q_r), (q_l, B, q_r) \to .\overleftarrow{c}(q_r, B, q_r)\}$
- $I_8 = \{(q_l, A, q_r) \to \overleftarrow{c}.(q_r, A, q_r), (q_r, A, q_r) \to .\overrightarrow{a}\}$

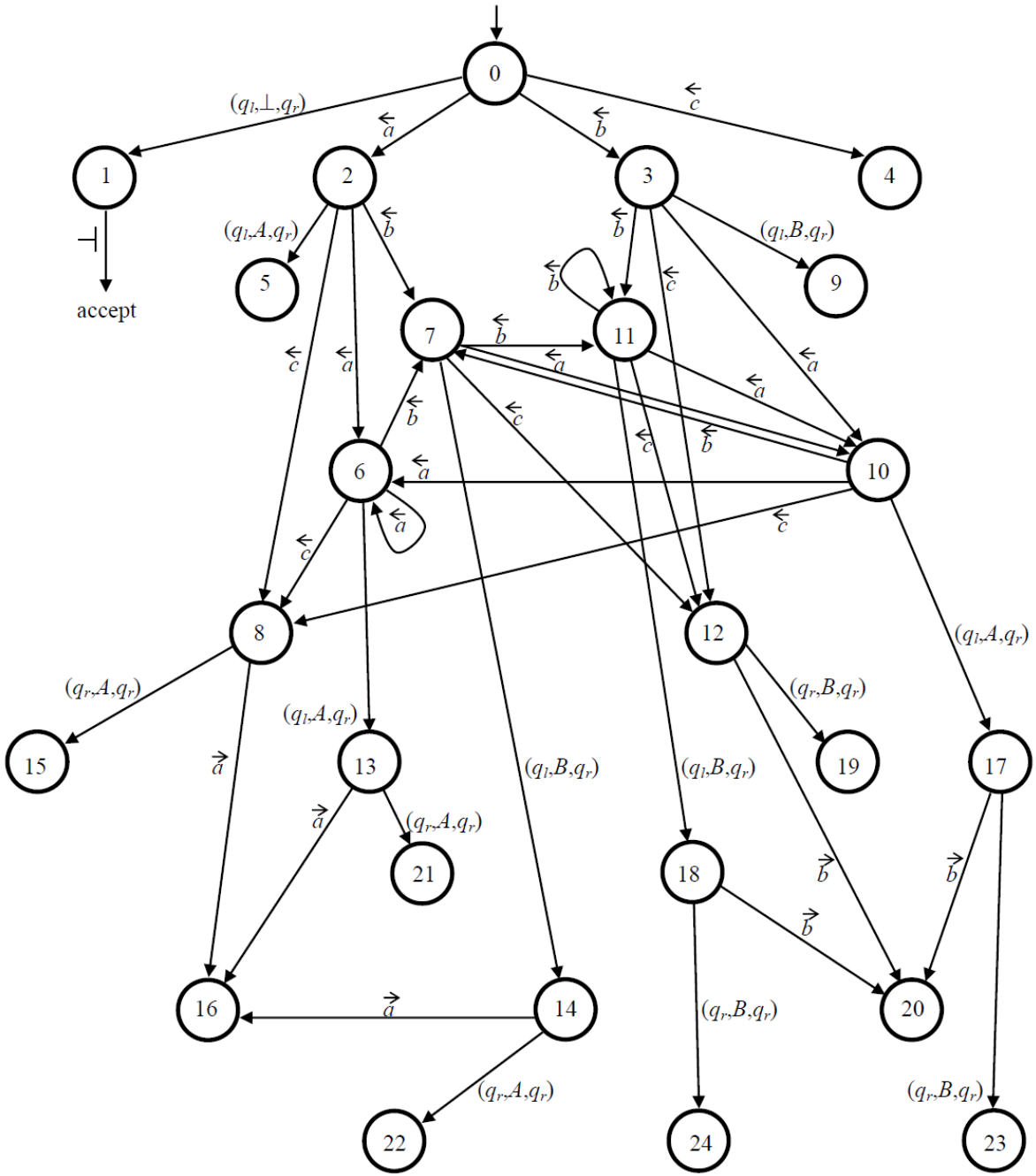
Example – LR-states

- $I_9 = \{(q_l, \bot, q_r) \to \overleftarrow{b}\,(q_l, B, q_r).\}$
- $I_{10} = \{(q_l, B, q_r) \to \overleftarrow{a}\,.(q_l, A, q_r)(q_r, B, q_r), (q_l, A, q_r) \to .\overleftarrow{a}\,(q_l, A, q_r)(q_r, A, q_r),$
  $(q_l, A, q_r) \to .\overleftarrow{b}\,(q_l, B, q_r)(q_r, A, q_r), (q_l, A, q_r) \to .\overleftarrow{c}\,(q_r, A, q_r)\}$
- $I_{11} = \{(q_l, B, q_r) \to \overleftarrow{b}\,.(q_l, B, q_r)(q_r, B, q_r), (q_l, B, q_r) \to .\overleftarrow{a}\,(q_l, A, q_r)(q_r, B, q_r),$
  $(q_l, B, q_r) \to .\overleftarrow{b}\,(q_l, B, q_r)(q_r, B, q_r), (q_l, B, q_r) \to .\overleftarrow{c}\,(q_r, B, q_r)\}$
- $I_{12} = \{(q_l, B, q_r) \to \overleftarrow{c}\,.(q_r, B, q_r), (q_r, B, q_r) \to .\overrightarrow{b}\}$
- $I_{13} = \{(q_l, A, q_r) \to \overleftarrow{a}\,(q_l, A, q_r).(q_r, A, q_r), (q_r, A, q_r) \to .\overrightarrow{a}\}$
- $I_{14} = \{(q_l, A, q_r) \to \overleftarrow{b}\,(q_l, B, q_r).(q_r, A, q_r), (q_r, A, q_r) \to .\overrightarrow{a}\}$
- $I_{15} = \{(q_l, A, q_r) \to \overleftarrow{c}\,(q_r, A, q_r).\}$
- $I_{16} = \{(q_r, A, q_r) \to \overrightarrow{a}\,.\}$
- $I_{17} = \{(q_l, B, q_r) \to \overleftarrow{a}\,(q_l, A, q_r).(q_r, B, q_r), (q_r, B, q_r) \to .\overrightarrow{b}\}$
- $I_{18} = \{(q_l, B, q_r) \to \overleftarrow{b}\,(q_l, B, q_r).(q_r, B, q_r), (q_r, B, q_r) \to .\overrightarrow{b}\}$

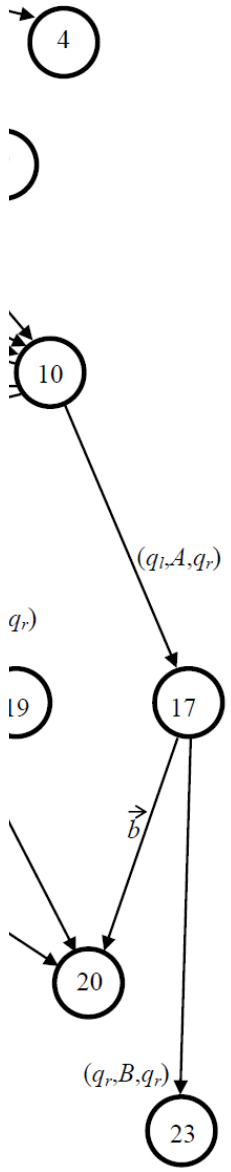
Example – LR-states

- $I_{19} = \{(q_l, B, q_r) \rightarrow \overleftarrow{c}\,(q_r, B, q_r).\}$
- $I_{20} = \{(q_r, B, q_r) \rightarrow \overrightarrow{b}.\}$
- $I_{21} = \{(q_l, A, q_r) \rightarrow \overleftarrow{a}\,(q_l, A, q_r)(q_r, A, q_r).\}$
- $I_{22} = \{(q_l, A, q_r) \rightarrow \overleftarrow{b}\,(q_l, B, q_r)(q_r, A, q_r).\}$
- $I_{23} = \{(q_l, B, q_r) \rightarrow \overleftarrow{a}\,(q_l, A, q_r)(q_r, B, q_r).\}$
- $I_{24} = \{(q_l, B, q_r) \rightarrow \overleftarrow{b}\,(q_l, B, q_r)(q_r, B, q_r).\}$

Example – LR-automaton



ACTION	$\overleftarrow{a}$	$\overleftarrow{b}$	$\overleftarrow{c}$	$\overrightarrow{d}$	$\overrightarrow{b}$	$\dashv$
0	SHIFT 2	SHIFT 3	SHIFT 4			
1						ACCEPT
2	SHIFT 6	SHIFT 7	SHIFT 8			
3	SHIFT 10	SHIFT 11	SHIFT 12			
4						REDUCE
5						REDUCE
6	SHIFT 6	SHIFT 7	SHIFT 8			
7	SHIFT 10	SHIFT 11	SHIFT 12			
8				SHIFT 16		
9						REDUCE
10	SHIFT 6	SHIFT 7	SHIFT 8			
11	SHIFT 10	SHIFT 11	SHIFT 12			
12					SHIFT 20	
13				SHIFT 16		
14				SHIFT 16		
15				REDUCE	REDUCE	REDUCE
16				REDUCE	REDUCE	REDUCE
17					SHIFT 20	
18					SHIFT 20	
19				REDUCE	REDUCE	REDUCE
20				REDUCE	REDUCE	REDUCE
21				REDUCE	REDUCE	REDUCE
22				REDUCE	REDUCE	REDUCE
23				REDUCE	REDUCE	REDUCE
24				REDUCE	REDUCE	REDUCE



Conclusion

- 2head PDA is a relatively new model defining a midly CS class of languages
- Its determinsitic counterpart is investigated.
- They are also characterized by controlled grammars.
- Closure and non-closure properties are shown.
- LR parsing is extended to det.2hPDA languages.

Thank you for your attention!