Approximation of weighted automata with storage

Tobias Denkinger

`tobias.denkinger@tu-dresden.de`

Institute of Theoretical Computer Science
Faculty of Computer Science
Technische Universität Dresden

GandALF, Rome, 2017-09-20

Language models in natural language processing

- regular grammars
- context-free grammars
- head grammars[1]
- multiple context-free grammars[2]

[1] or yields of tree adjoining grammars
[2] or linear context-free rewriting systems

Language models in natural language processing

parsing complexities:

- regular grammars $\mathcal{O}(n)$
- context-free grammars (binarised) $\mathcal{O}(n^3)$
- head grammars[1] $\mathcal{O}(n^6)$
- multiple context-free grammars[2] (binarised, fan-out k) $\mathcal{O}(n^{3k})$

[1] or yields of tree adjoining grammars
[2] or linear context-free rewriting systems

Language models in natural language processing

parsing complexities:

- regular grammars $\mathcal{O}(n)$
- context-free grammars (binarised) $\mathcal{O}(n^3)$
- head grammars[1] $\mathcal{O}(n^6)$
- multiple context-free grammars[2] (binarised, fan-out k) $\mathcal{O}(n^{3k})$

problems:

- high parsing complexities

[1] or yields of tree adjoining grammars
[2] or linear context-free rewriting systems

Language models in natural language processing

parsing complexities:

- regular grammars $\mathcal{O}(n)$
- context-free grammars (binarised) $\mathcal{O}(n^3)$
- head grammars[1] $\mathcal{O}(n^6)$
- multiple context-free grammars[2] (binarised, fan-out k) $\mathcal{O}(n^{3k})$

problems:

- high parsing complexities $\longrightarrow$ use *approximation*

[1] or yields of tree adjoining grammars
[2] or linear context-free rewriting systems

Language models in natural language processing

parsing complexities:

- regular grammars $\mathcal{O}(n)$
- context-free grammars (binarised) $\mathcal{O}(n^3)$
- head grammars[1] $\mathcal{O}(n^6)$
- multiple context-free grammars[2] (binarised, fan-out k) $\mathcal{O}(n^{3k})$

problems:

- high parsing complexities → use *approximation*
- natural languages are ambiguous

[1] or yields of tree adjoining grammars
[2] or linear context-free rewriting systems

Language models in natural language processing

parsing complexities:

- regular grammars $\mathcal{O}(n)$
- context-free grammars (binarised) $\mathcal{O}(n^3)$
- head grammars[1] $\mathcal{O}(n^6)$
- multiple context-free grammars[2] (binarised, fan-out k) $\mathcal{O}(n^{3k})$

problems:

- high parsing complexities → use *approximation*
- natural languages are ambiguous → use *weights* (semirings)

[1] or yields of tree adjoining grammars
[2] or linear context-free rewriting systems

Language models in natural language processing

parsing complexities:

- regular grammars $\mathcal{O}(n)$
- context-free grammars (binarised) $\mathcal{O}(n^3)$
- head grammars[1] $\mathcal{O}(n^6)$
- multiple context-free grammars[2] (binarised, fan-out k) $\mathcal{O}(n^{3k})$

problems:

- high parsing complexities → use *approximation*
- natural languages are ambiguous → use *weights* (semirings)
- lots of different models

[1] or yields of tree adjoining grammars

[2] or linear context-free rewriting systems

Language models in natural language processing

parsing complexities:

- regular grammars $\mathcal{O}(n)$
  ≡ finite state automata
- context-free grammars (binarised) $\mathcal{O}(n^3)$
  ≡ pushdown automata [1, 2]
- head grammars[1] $\mathcal{O}(n^6)$
  ≡ embedded pushdown automata [3]
- multiple context-free grammars[2] (binarised, fan-out k) $\mathcal{O}(n^{3k})$
  ≡ k-restricted tree-stack automata [4]

problems:

- high parsing complexities → use *approximation*
- natural languages are ambiguous → use *weights* (semirings)
- lots of different models → use *automata with storage*

[1] or yields of tree adjoining grammars
[2] or linear context-free rewriting systems

Outline

1. Weighted automata with storage
2. Approximation of storage
3. Approximation of weighted automata with storage
4. Coarse-to-fine n-best parsing

Outline

1. Weighted automata with storage
2. Approximation of storage
3. Approximation of weighted automata with storage
4. Coarse-to-fine n-best parsing

Storage

$S = (C, P, R, c_\mathsf{i})$

Example: Count

Storage

$S = (C, P, R, c_\mathsf{i})$

C set… *configurations*

Example: Count

$C = \mathbb{N}$

Storage

$S = (C, P, R, c_{\mathsf{i}})$

C set… *configurations*

$P \subseteq \mathcal{P}(C)$… *predicates*

Example: Count

$C = \mathbb{N}$

$P = \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\}$

Storage

$S = (C, P, R, c_{\mathsf{i}})$

C set... *configurations*

$P \subseteq \mathcal{P}(C)$... *predicates*

$R \subseteq \mathcal{P}(C \times C)$... *instructions*

every $r \in R$ finitely non-det.

($\forall c \in C\colon r(c)$ is finite)

Example: Count

$C = \mathbb{N}$

$P = \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\}$

$R = \{\text{inc}, \text{dec}\}$

$\text{inc} = \{(n, n+1) \mid n \in \mathbb{N}\}$

$\text{dec} = \text{inc}^{-1}$

Storage

$S = (C, P, R, c_\mathsf{i})$

C set… *configurations*

$P \subseteq \mathcal{P}(C)$… *predicates*

$R \subseteq \mathcal{P}(C \times C)$… *instructions*

every $r \in R$ finitely non-det.

($\forall c \in C\colon r(c)$ is finite)

$c_\mathsf{i} \in C$… *initial configuration*

Example: Count

$C = \mathbb{N}$

$P = \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\}$

$R = \{\mathrm{inc}, \mathrm{dec}\}$

$\mathrm{inc} = \{(n, n+1) \mid n \in \mathbb{N}\}$

$\mathrm{dec} = \mathrm{inc}^{-1}$

$c_\mathsf{i} = 0$

Storage

$S = (C, P, R, c_\mathsf{i})$

C set… *configurations*

$P \subseteq \mathcal{P}(C)$… *predicates*

$R \subseteq \mathcal{P}(C \times C)$… *instructions*

every $r \in R$ finitely non-det.

($\forall c \in C\colon r(c)$ is finite)

$c_\mathsf{i} \in C$… *initial configuration*

Example: Count

$C = \mathbb{N}$

$P = \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\}$

$R = \{\mathrm{inc}, \mathrm{dec}\}$

$\mathrm{inc} = \{(n, n+1) \mid n \in \mathbb{N}\}$

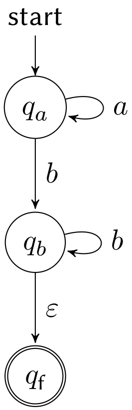
$\mathrm{dec} = \mathrm{inc}^{-1}$

$c_\mathsf{i} = 0$

instances:
- Goldstine's [5] data storage: $P = \{C\}$
- Engelfriet's [6, 7] storage types: $R \subseteq C \dashrightarrow C$

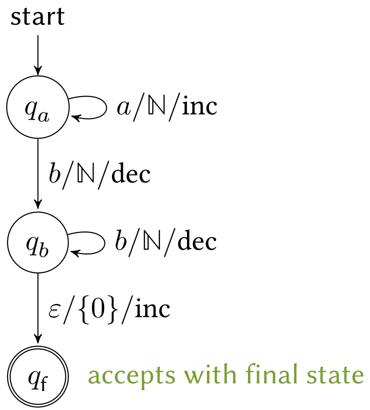
ℕ-weighted automaton with Count-storage

automaton $\mathcal{M}$:



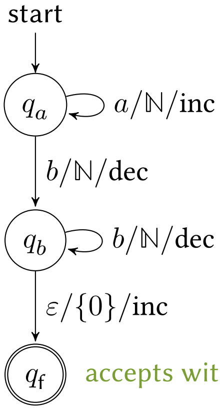
$\mathbb{N}$-weighted automaton with Count-storage

automaton $\mathcal{M}$:



$\mathbb{N}$-weighted automaton with Count-storage

automaton $\mathcal{M}$:

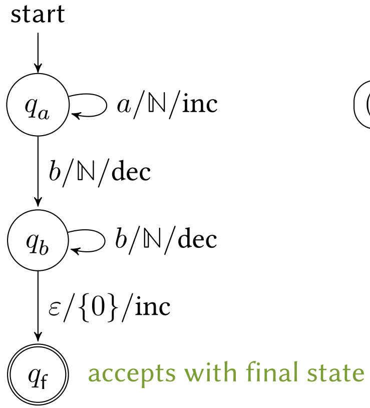


accepts with final state

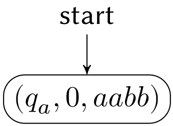
$$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$$

$\mathbb{N}$-weighted automaton with Count-storage

automaton $\mathcal{M}$:



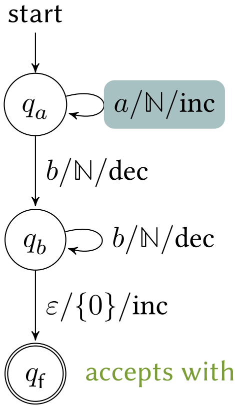
run of $\mathcal{M}$ on $aabb$:



$$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$$

$\mathbb{N}$-weighted automaton with Count-storage

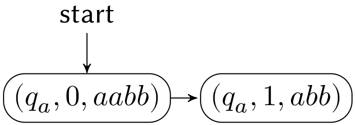
automaton $\mathcal{M}$:



accepts with final state

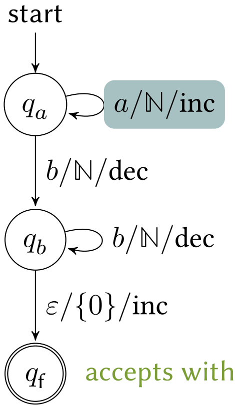
$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$

run of $\mathcal{M}$ on $aabb$:



$\mathbb{N}$-weighted automaton with Count-storage

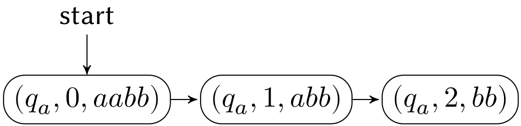
automaton $\mathcal{M}$:



accepts with final state

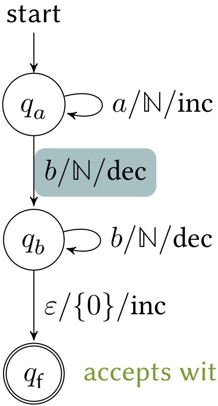
$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$

run of $\mathcal{M}$ on $aabb$:



ℕ-weighted automaton with Count-storage

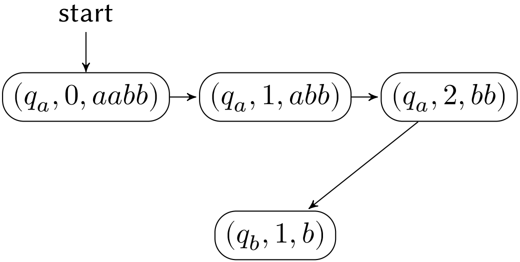
automaton $\mathcal{M}$:



accepts with final state

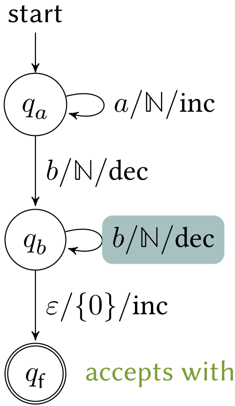
$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$

run of $\mathcal{M}$ on $aabb$:



ℕ-weighted automaton with Count-storage

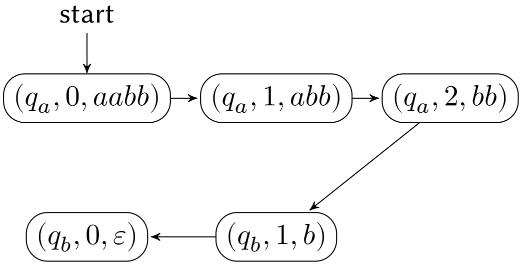
automaton $\mathcal{M}$:



accepts with final state

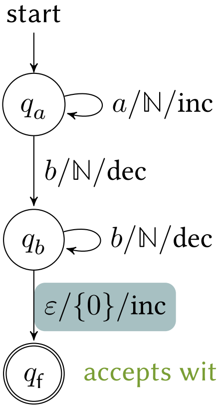
$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$

run of $\mathcal{M}$ on $aabb$:



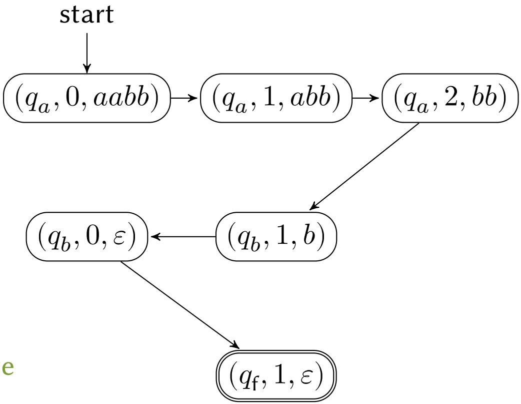
$\mathbb{N}$-weighted automaton with Count-storage

automaton $\mathcal{M}$:



accepts with final state

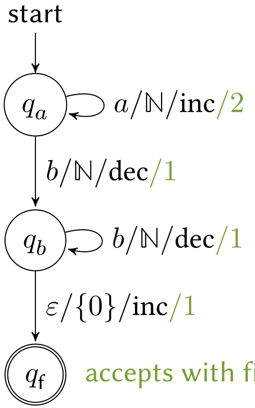
run of $\mathcal{M}$ on $aabb$:



$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$ $\quad aabb \in L(\mathcal{M})$

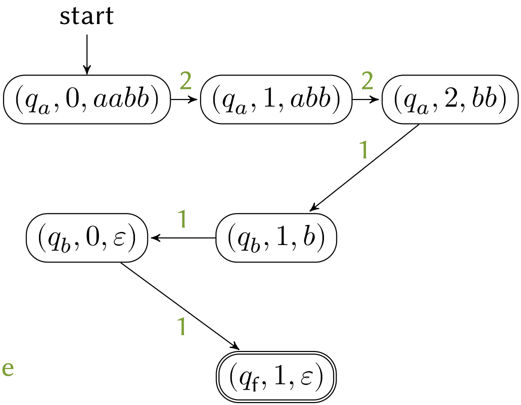
$\mathbb{N}$-weighted automaton with Count-storage

automaton $\mathcal{M}$:



accepts with final state

run of $\mathcal{M}$ on $aabb$: $(\mathbb{N}, +, \cdot, 0, 1)$



$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$ $\quad aabb \in L(\mathcal{M})$ $\quad [\![\mathcal{M}]\!](aabb) = 4$

$[\![\mathcal{M}]\!](a^n b^n) = 2^n$

Outline

Approximation of storage

given: $S = (C, P, R, c_\mathsf{i})$, *approximation strategy* $A\colon C \dashrightarrow C'$

target: *approximation storage* $A(S)$

Approximation of storage

given: $S = (C, P, R, c_\mathsf{i})$, *approximation strategy* $A\colon C \dashrightarrow C'$

target: *approximation storage* $A(S)$

- configurations: apply A

Approximation of storage

given: $S = (C, P, R, c_\mathsf{i})$, *approximation strategy* $A\colon C \dashrightarrow C'$

target: *approximation storage* $A(S)$

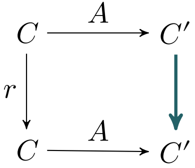
- configurations: apply A
- predicates: apply A pointwise

Approximation of storage

given: $S = (C, P, R, c_\mathrm{i})$, *approximation strategy* $A\colon C \dashrightarrow C'$

target: *approximation storage* $A(S)$

- configurations: apply A
- predicates: apply A pointwise
- instructions:



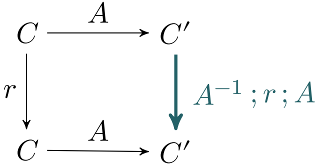
arrows denote binary relations

Approximation of storage

given: $S = (C, P, R, c_{\mathrm{i}})$, *approximation strategy* $A\colon C \dashrightarrow C'$

target: *approximation storage* $A(S)$

- configurations: apply A
- predicates: apply A pointwise
- instructions:



arrows denote binary relations

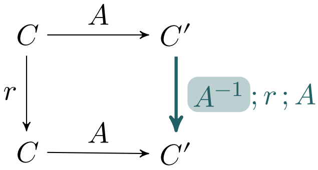
Approximation of storage

given: $S = (C, P, R, c_{\mathsf{i}})$, *approximation strategy* $A\colon C \dashrightarrow C'$

target: *approximation storage* $A(S)$

- configurations: apply A
- predicates: apply A pointwise
- instructions:

A^{-1} may introduce non-determinism



arrows denote binary relations

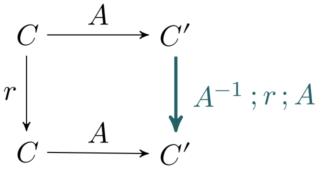
Approximation of storage

given: $S = (C, P, R, c_{\mathsf{i}})$, *approximation strategy* $A\colon C \dashrightarrow C'$

target: *approximation storage* $A(S)$

- configurations: apply A
- predicates: apply A pointwise
- instructions:

A^{-1} may introduce non-determinism



arrows denote binary relations

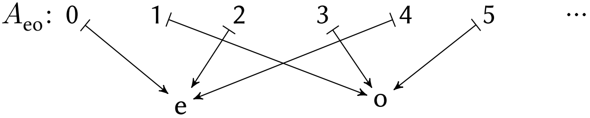
$A(S)$ is only defined if $A^{-1}\,;r\,;A$ is finitely non-det. for each r.

An approximation of Count

$$\mathrm{Count} = \big(\mathbb{N},\ \ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \ \{\mathrm{inc}, \mathrm{dec}\},\ \ 0\big)$$

An approximation of Count

$\text{Count} = (\mathbb{N},\ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \{\text{inc}, \text{dec}\},\ 0)$

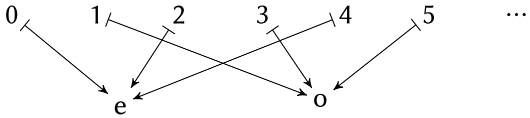


$A_{\text{eo}}(\text{Count}) = (\{\text{e}, \text{o}\},\qquad\qquad,\qquad\qquad,\ \text{e})$

An approximation of Count

$\text{Count} = \big(\mathbb{N},\ \ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \ \{\text{inc}, \text{dec}\},\ \ 0\big)$

A_{eo}:



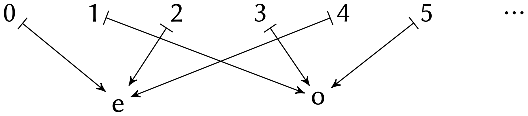
- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\}$

$A_{\text{eo}}(\text{Count}) = \big(\{\text{e}, \text{o}\},\qquad\qquad ,\qquad\qquad ,\ \ \text{e}\big)$

An approximation of Count

$\text{Count} = (\mathbb{N},\ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \{\text{inc}, \text{dec}\},\ 0)$

A_{eo}:



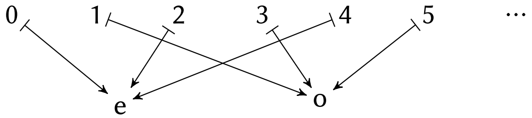
- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\} = A_{\text{eo}}(\mathbb{N} \setminus \{0\})$

$A_{\text{eo}}(\text{Count}) = (\{\text{e}, \text{o}\},\qquad\qquad , \qquad\qquad ,\ \text{e})$

An approximation of Count

$$\text{Count} = \big(\mathbb{N},\ \ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \ \{\text{inc}, \text{dec}\},\ \ 0\big)$$

A_{eo}:

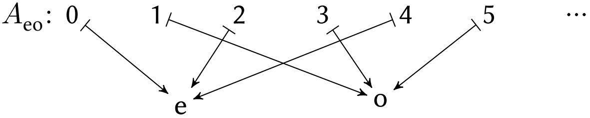


- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\} = A_{\text{eo}}(\mathbb{N} \setminus \{0\})$ $\qquad A_{\text{eo}}(\{0\}) = \{\text{e}\}$

$$A_{\text{eo}}(\text{Count}) = \big(\{\text{e}, \text{o}\},\ \ \{\{\text{e}, \text{o}\}, \{\text{e}\}\},\ \qquad\qquad ,\ \ \text{e}\big)$$

An approximation of Count

$\text{Count} = \big(\mathbb{N},\ \ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \ \{\text{inc}, \text{dec}\},\ \ 0\big)$



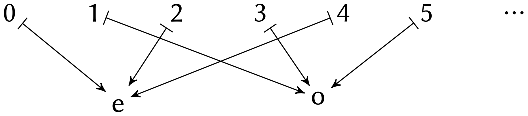
- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\} = A_{\text{eo}}(\mathbb{N} \setminus \{0\})$ $\qquad A_{\text{eo}}(\{0\}) = \{\text{e}\}$
- $A_{\text{eo}}(\text{inc}) = \{ \qquad\qquad \}$

$A_{\text{eo}}(\text{Count}) = \big(\{\text{e}, \text{o}\},\ \ \{\{\text{e}, \text{o}\}, \{\text{e}\}\}, \qquad\qquad ,\ \ \text{e}\big)$

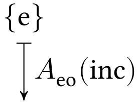
An approximation of Count

$\text{Count} = \big(\mathbb{N},\ \ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \ \{\text{inc}, \text{dec}\},\ \ 0\big)$

A_{eo}:



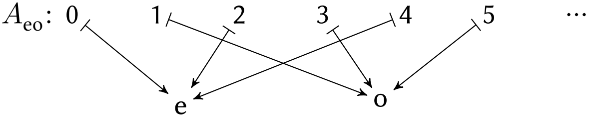
- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\} = A_{\text{eo}}(\mathbb{N} \setminus \{0\})$ $\qquad A_{\text{eo}}(\{0\}) = \{\text{e}\}$
- $A_{\text{eo}}(\text{inc}) = \{ \qquad\qquad \}$



$A_{\text{eo}}(\text{Count}) = \big(\{\text{e}, \text{o}\},\ \ \{\{\text{e}, \text{o}\}, \{\text{e}\}\}, \qquad\qquad ,\ \ \text{e}\big)$

An approximation of Count

$\text{Count} = \big(\mathbb{N},\ \ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \ \{\text{inc}, \text{dec}\},\ \ 0\big)$



- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\} = A_{\text{eo}}(\mathbb{N} \setminus \{0\})$ $\qquad A_{\text{eo}}(\{0\}) = \{\text{e}\}$
- $A_{\text{eo}}(\text{inc}) = \{ \qquad\qquad \}$

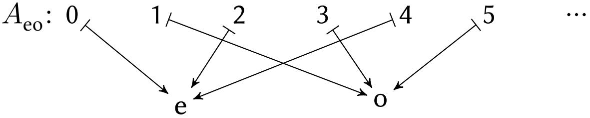
$$\{0, 2, 4, \dots\} \xmapsto{A_{\text{eo}}} \{\text{e}\}$$

$$\{\text{e}\} \ \big\downarrow A_{\text{eo}}(\text{inc})$$

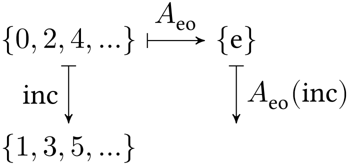
$A_{\text{eo}}(\text{Count}) = \big(\{\text{e}, \text{o}\},\ \ \{\{\text{e}, \text{o}\}, \{\text{e}\}\}, \qquad\qquad ,\ \ \text{e}\big)$

An approximation of Count

$\text{Count} = (\mathbb{N},\ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \{\text{inc}, \text{dec}\},\ 0)$



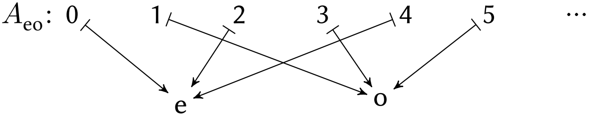
- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\} = A_{\text{eo}}(\mathbb{N} \setminus \{0\}) \qquad A_{\text{eo}}(\{0\}) = \{\text{e}\}$
- $A_{\text{eo}}(\text{inc}) = \{ \qquad\qquad \}$



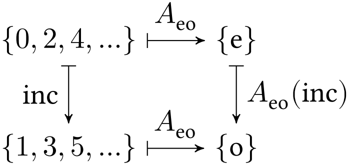
$A_{\text{eo}}(\text{Count}) = (\{\text{e}, \text{o}\},\ \{\{\text{e}, \text{o}\}, \{\text{e}\}\}, \qquad\qquad ,\ \text{e})$

An approximation of Count

$\text{Count} = (\mathbb{N},\ \ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \ \{\text{inc}, \text{dec}\},\ \ 0)$



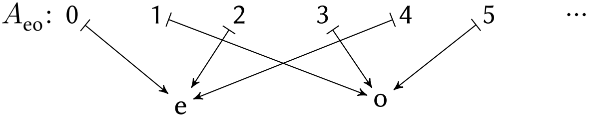
- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\} = A_{\text{eo}}(\mathbb{N} \setminus \{0\})$ $\qquad A_{\text{eo}}(\{0\}) = \{\text{e}\}$
- $A_{\text{eo}}(\text{inc}) = \{(\text{e}, \text{o}) \qquad \}$



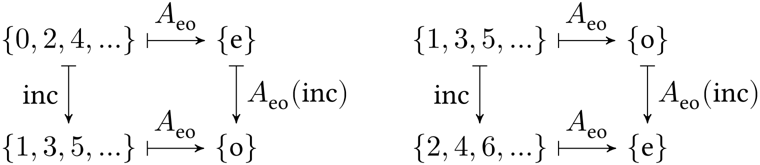
$A_{\text{eo}}(\text{Count}) = (\{\text{e}, \text{o}\},\ \ \{\{\text{e}, \text{o}\}, \{\text{e}\}\}, \qquad\qquad ,\ \ \text{e})$

An approximation of Count

$\mathrm{Count} = (\mathbb{N},\ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \{\mathrm{inc}, \mathrm{dec}\},\ 0)$



- $A_{\mathrm{eo}}(\mathbb{N}) = \{\mathrm{e}, \mathrm{o}\} = A_{\mathrm{eo}}(\mathbb{N} \setminus \{0\})$ $\qquad A_{\mathrm{eo}}(\{0\}) = \{\mathrm{e}\}$
- $A_{\mathrm{eo}}(\mathrm{inc}) = \{(\mathrm{e}, \mathrm{o}), (\mathrm{o}, \mathrm{e})\}$

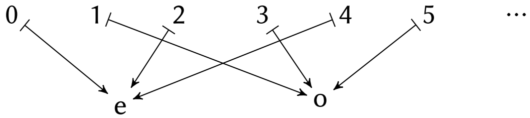


$A_{\mathrm{eo}}(\mathrm{Count}) = (\{\mathrm{e}, \mathrm{o}\},\ \{\{\mathrm{e}, \mathrm{o}\}, \{\mathrm{e}\}\},\ \qquad\qquad ,\ \mathrm{e})$

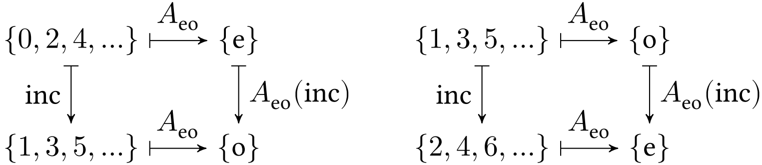
An approximation of Count

$$\text{Count} = (\mathbb{N},\ \ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \ \{\text{inc}, \text{dec}\},\ \ 0)$$

A_{eo}:



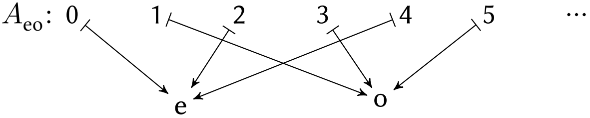
- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\} = A_{\text{eo}}(\mathbb{N} \setminus \{0\})$ $\qquad A_{\text{eo}}(\{0\}) = \{\text{e}\}$
- $A_{\text{eo}}(\text{inc}) = \{(\text{e}, \text{o}), (\text{o}, \text{e})\} = A_{\text{eo}}(\text{dec})$



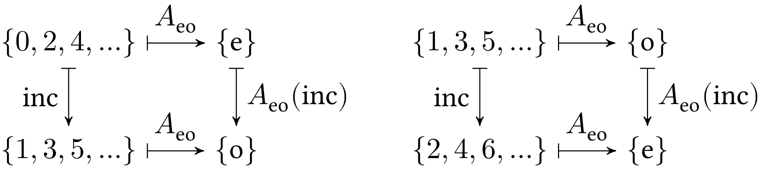
$$A_{\text{eo}}(\text{Count}) = (\{\text{e}, \text{o}\},\ \ \{\{\text{e}, \text{o}\}, \{\text{e}\}\},\qquad\qquad ,\ \ \text{e})$$

An approximation of Count

$$\text{Count} = (\mathbb{N},\ \ \{\mathbb{N}, \{0\}, \mathbb{N} \setminus \{0\}\},\ \ \{\text{inc}, \text{dec}\},\ \ 0)$$



- $A_{\text{eo}}(\mathbb{N}) = \{\text{e}, \text{o}\} = A_{\text{eo}}(\mathbb{N} \setminus \{0\}) \qquad A_{\text{eo}}(\{0\}) = \{\text{e}\}$
- $A_{\text{eo}}(\text{inc}) = \underbrace{\{(\text{e}, \text{o}), (\text{o}, \text{e})\}}_{\text{toggle}} = A_{\text{eo}}(\text{dec})$



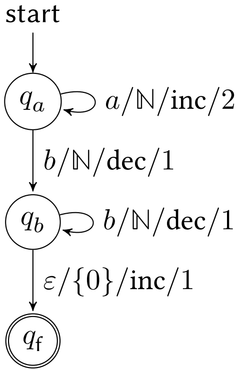
$$A_{\text{eo}}(\text{Count}) = (\{\text{e}, \text{o}\},\ \ \{\{\text{e}, \text{o}\}, \{\text{e}\}\},\ \ \{\text{toggle}\},\ \ \text{e})$$

Outline

1. Weighted automata with storage
2. Approximation of storage
3. Approximation of weighted automata with storage
4. Coarse-to-fine n-best parsing

Approximation of $\mathbb{N}$-weighted aut. with Count-storage

automaton $\mathcal{M}$:

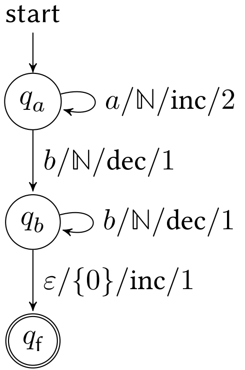


$$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$$

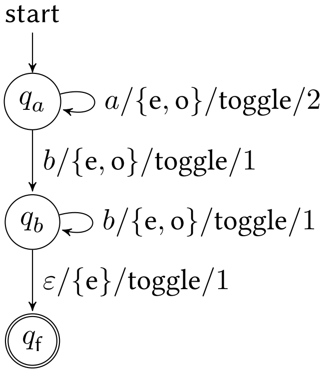
$$[\![\mathcal{M}]\!](a^n b^n) = 2^n$$

Approximation of ℕ-weighted aut. with Count-storage

automaton $\mathcal{M}$:



automaton $A_{\mathrm{eo}}(\mathcal{M})$:

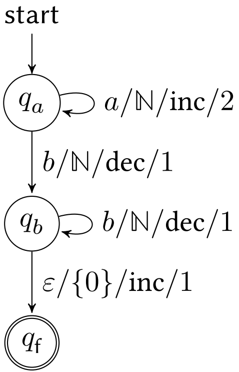


$$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$$

$$[\![\mathcal{M}]\!](a^n b^n) = 2^n$$

Approximation of $\mathbb{N}$-weighted aut. with Count-storage

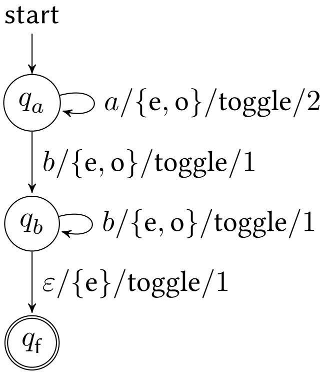
automaton $\mathcal{M}$:



$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$

$[\![\mathcal{M}]\!](a^n b^n) = 2^n$

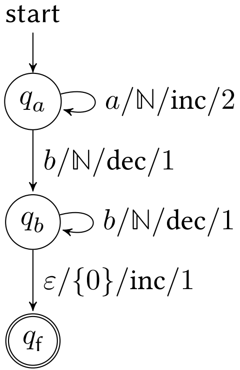
automaton $A_{\text{eo}}(\mathcal{M})$:



$$L(A_{\text{eo}}(\mathcal{M})) = \{a^m b^n \mid m \geq 0, n \geq 1, m \equiv n \mod 2\}$$

Approximation of $\mathbb{N}$-weighted aut. with Count-storage

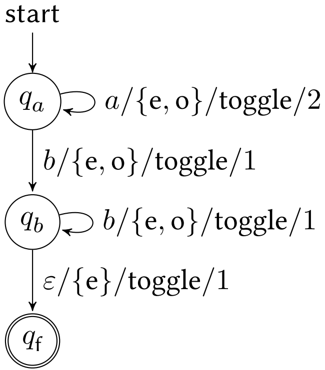
automaton $\mathcal{M}$:



$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$

$[\![\mathcal{M}]\!](a^n b^n) = 2^n$

automaton $A_{eo}(\mathcal{M})$:



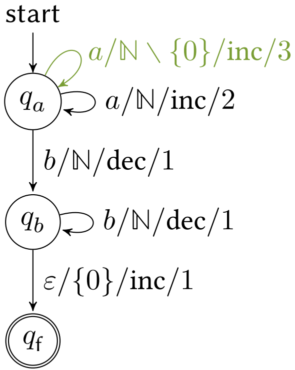
$$L(A_{eo}(\mathcal{M})) = \{a^m b^n \mid m \geq 0, n \geq 1, m \equiv n \mod 2\}$$

$$[\![A_{eo}(\mathcal{M})]\!](a^m b^n) = 2^m$$

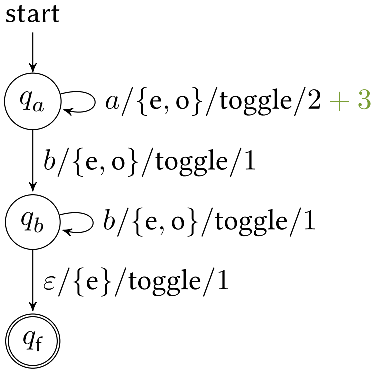
$$m \equiv n \mod 2$$

Approximation of $\mathbb{N}$-weighted aut. with Count-storage

automaton $\mathcal{M}$:



automaton $A_{\mathrm{eo}}(\mathcal{M})$:



$$L(\mathcal{M}) = \{a^n b^n \mid n \geq 1\}$$

$$L(A_{\mathrm{eo}}(\mathcal{M})) = \{a^m b^n \mid m \geq 0, n \geq 1, m \equiv n \mod 2\}$$

$$[\![\mathcal{M}]\!](a^n b^n) = 2 \cdot (2+3)^{n-1}$$

$$[\![A_{\mathrm{eo}}(\mathcal{M})]\!](a^m b^n) = (2+3)^m \quad m \equiv n \mod 2$$

Approximation of weighted automata with storage

Theorem (unweighted).

Let $\mathcal{M}$ be an (S, Σ)-automaton and A be an S-proper approximation strategy.

- If A is *total*, then $L(A(\mathcal{M})) \supseteq L(\mathcal{M})$.
- If A is *injective*, then $L(A(\mathcal{M})) \subseteq L(\mathcal{M})$.

Approximation of weighted automata with storage

Theorem (unweighted).

Let $\mathcal{M}$ be an (S, Σ)-automaton and A be an S-proper approximation strategy.

- If A is *total*, then $L(A(\mathcal{M})) \supseteq L(\mathcal{M})$.
- If A is *injective*, then $L(A(\mathcal{M})) \subseteq L(\mathcal{M})$.

Theorem (weighted).

Let $\mathcal{M}$ be an (S, Σ, K)-automaton, A be an S-proper approximation strategy, $\leq$ be a partial order on K, and K be positively $\leq$-ordered.

- If A is *total*, then $[\![A(\mathcal{M})]\!](w) \geq [\![\mathcal{M}]\!](w)$ for every $w \in \Sigma^*$.
- If A is *injective*, then $[\![A(\mathcal{M})]\!](w) \leq [\![\mathcal{M}]\!](w)$ for every $w \in \Sigma^*$.

Approximation of weighted automata with storage

Theorem (unweighted).

Let $\mathcal{M}$ be an (S, Σ)-automaton and A be an S-proper approximation strategy.

- If A is *total*, then $L(A(\mathcal{M})) \supseteq L(\mathcal{M})$.
- If A is *injective*, then $L(A(\mathcal{M})) \subseteq L(\mathcal{M})$.

Theorem (weighted).

Let $\mathcal{M}$ be an (S, Σ, K)-automaton, A be an S-proper approximation strategy, $\leq$ be a partial order on K, and K be positively $\leq$-ordered.

- If A is *total*, then $[\![A(\mathcal{M})]\!](w) \geq [\![\mathcal{M}]\!](w)$ for every $w \in \Sigma^*$.
- If A is *injective*, then $[\![A(\mathcal{M})]\!](w) \leq [\![\mathcal{M}]\!](w)$ for every $w \in \Sigma^*$.

Theorem (unweighted) is a corollary of Theorem (weighted)

[by setting $K = \mathbb{B}$]

Outline

1. Weighted automata with storage
2. Approximation of storage
3. Approximation of weighted automata with storage
4. Coarse-to-fine n-best parsing

n-best parsing (for automata)

parsing

Input: an automaton $\mathcal{M}$, a word w

Output: the set of all runs of $\mathcal{M}$ on w

n-best parsing (for automata)

parsing

Input: an automaton $\mathcal{M}$, a word w

Output: the set of all runs of $\mathcal{M}$ on w

n-best parsing

Input: a K-weighted automaton $\mathcal{M}$, a word w, a number n

Output: a sequence of n best (w.r.t. weight) runs of $\mathcal{M}$ on w

Coarse-to-fine n-best parsing

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$

2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

Coarse-to-fine n-best parsing

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing

Input: a K-weighted automaton $\mathcal{M}$, a word w, a number n, proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w

Coarse-to-fine n-best parsing

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$

2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing

Input: a K-weighted automaton $\mathcal{M}$, a word w, a number n, proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w

$$(\mathcal{M}, w) \xrightarrow{\text{2-best parse}}$$

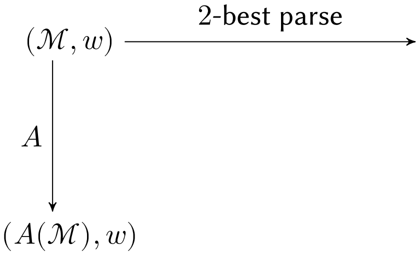
Coarse-to-fine n-best parsing

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing

Input: a K-weighted automaton $\mathcal{M}$, a word w, a number n,
proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



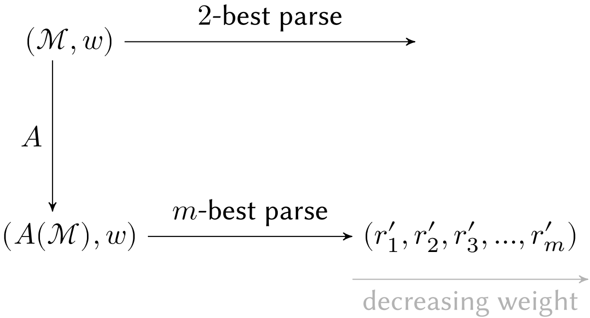
Coarse-to-fine n-best parsing

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing

Input: a K-weighted automaton $\mathcal{M}$, a word w, a number n,
proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



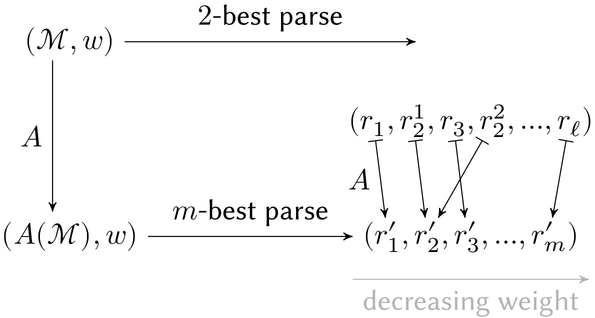
Coarse-to-fine n-best parsing

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing

Input: a K-weighted automaton $\mathcal{M}$, a word w, a number n, proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



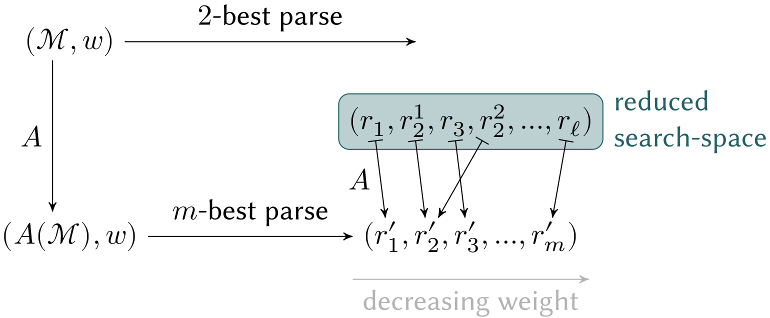
Coarse-to-fine n-best parsing

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing

Input: a K-weighted automaton $\mathcal{M}$, a word w, a number n, proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



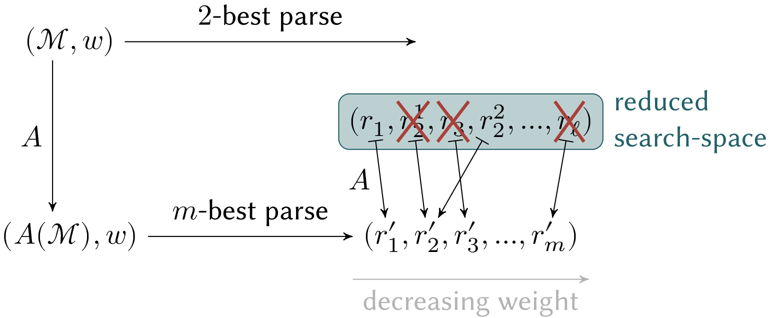
Coarse-to-fine n-best parsing

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing

Input: a K-weighted automaton $\mathcal{M}$, a word w, a number n, proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



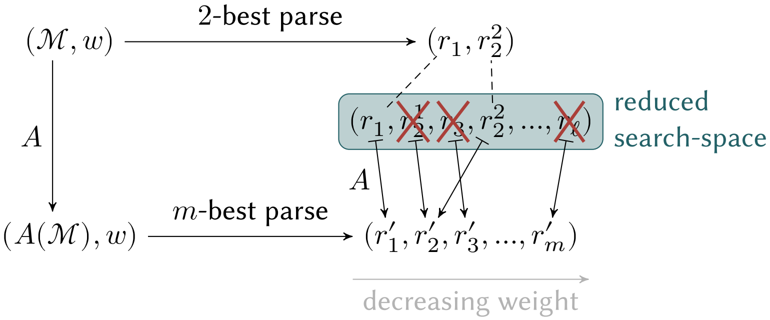
Coarse-to-fine n-best parsing

Idea: 1. recognise word with a *superset* approximation $A(\mathcal{M})$
2. use runs of $A(\mathcal{M})$ to **reduce search space** for runs of $\mathcal{M}$

coarse-to-fine n-best parsing

Input: a K-weighted automaton $\mathcal{M}$, a word w, a number n, proper *total* approximation strategy A

Output: a sequence of n best runs of $\mathcal{M}$ on w



Summary

- Approximation strategies are modelled as partial functions.

Summary

- Approximation strategies are modelled as partial functions.
- Properties of the approximation strategy imply properties of the approximation process.

Summary

- Approximation strategies are modelled as partial functions.
- Properties of the approximation strategy imply properties of the approximation process.
- We presented a generic coarse-to-fine n-best parser.

Summary

- Approximation strategies are modelled as partial functions.
- Properties of the approximation strategy imply properties of the approximation process.
- We presented a generic coarse-to-fine n-best parser.

Thank you for your attention.

References

[1] N. Chomsky. “Formal properties of grammars”. 1962.

[2] M. P. Schützenberger. “On context-free languages and push-down automata”. 1963.

[3] K. Vijay-Shanker. “A study of tree adjoining grammars”. PhD thesis. University of Pennsylvania, 1988.

[4] T. Denkinger. “An Automata Characterisation for Multiple Context-Free Languages”. 2016.

[5] J. Goldstine. “A rational theory of AFLs”. 1979.

[6] J. Engelfriet. *Context-free grammars with storage.* Tech. rep. Leiden University, 1986.

[7] L. Herrmann and H. Vogler. “A Chomsky-Schützenberger Theorem for Weighted Automata with Storage”. 2015.