M-monoid parsing and reduct generation

Richard Mörbitz

13th March 2018

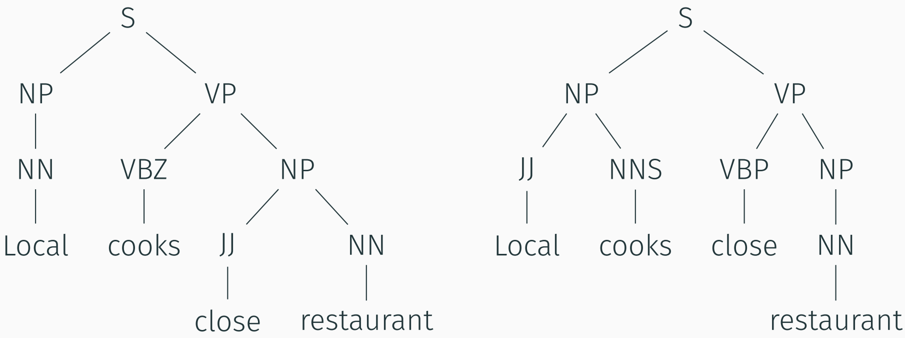
Introduction

Ambiguity in natural language

Local cooks close restaurant.

Ambiguity in natural language

Local cooks close restaurant.



Definitions

M-monoid

Definition (Multioperator monoid)

An M-monoid is an algebraic structure $(S, \oplus, 0, \Omega)$, such that

- $(S, \oplus, 0)$ is a commutative monoid,
- Ω is a set of operations on S such that $\forall \omega \in \Omega : \omega(\dots, 0, \dots) = 0$
- $0^k \in \Omega$ for all $k \in \mathbb{N}$, $0^k : S^k \to S$ such that $0^k(s_1, \dots, s_k) = 0$, and
- Ω distributes over $\oplus$

S is *complete* if the infinitary sum $\sum^{\oplus}$ exists.

Viterbi M-monoid

$(\mathbb{V}, \sum^{\max})$, where

- $\mathbb{V} = (\mathbb{R}_0^1, \max, 0, \Omega.)$
- ${\sum^{\max}}_{i \in I} s_i = \sup\{s_i \mid i \in I\}$

Operations in Ω.

$$\omega_a : (\mathbb{R}_0^1)^k \to \mathbb{R}_0^1$$
$$(s_1, \ldots, s_k) \mapsto a \cdot s_1 \cdot \ldots \cdot s_k$$

$(k \in \mathbb{N}, a \in \mathbb{R}_0^1)$

Definition (LCFRS)

Let Δ be a finite set. An LCFRS over Δ is a tuple $G = (N, \Sigma, Z, R)$, where

- N is a finite $\mathbb{N}$-sorted set (nonterminals),
- Σ is a finite $(\mathbb{N}^* \times \mathbb{N})$-sorted set (terminals) of the form $\langle \text{imagine}, x_1^{(1)} x_1^{(2)}, x_2^{(1)} \text{staff}, \varepsilon \rangle$ (sort $(2, 4)$),
- $Z \in N_1$ (initial nonterminal), and
- R is a finite ranked alphabet (rules) of the form $A \to \langle x_1^{(1)} x_1^{(3)}, x_1^{(2)} x_2^{(1)} \rangle (A_1, A_2, A_3)$ (rank 3).

Example

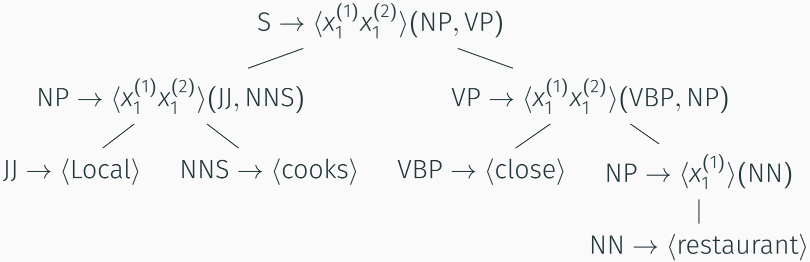
$\Delta = \{\text{Local}, \text{cooks}, \text{close}, \text{restaurant}\}$

$G = (N, \Sigma, Z, R)$, where

- $N = \{\text{S}, \text{NP}, \text{VP}, \text{NN}, \text{NNS}, \text{VBZ}, \text{VBP}, \text{JJ}\}$
- $\Sigma = \{\langle\text{Local}\rangle, \langle\text{cooks}\rangle, \langle\text{close}\rangle, \langle\text{restaurant}\rangle, \langle x_1^{(1)}\rangle, \langle x_1^{(1)} x_1^{(2)}\rangle\}$
- $Z = \text{S}$
- $R \supset \{\text{S} \rightarrow \langle x_1^{(1)} x_2^{(2)}\rangle(\text{NP VP}), \text{NP} \rightarrow \langle x_1^{(1)}\rangle(\text{NN}), \text{NN} \rightarrow \langle\text{Local}\rangle\}$

Abstract syntax trees

AST tree $d \in T_R$ such that for each $p \in \text{pos}(d)$:
if $d(p) = (A \rightarrow \sigma(A_1, \dots A_k))$,
then for each $i \in \{1, \dots, k\}$ the left-hand side of $d(pi)$ is A_i.



Definition (Range concatenation grammar)

An RCG is a tuple $G = (N, \Delta, Z, R)$, where

- N is a finite $\mathbb{N}$-sorted set (nonterminals),
- Δ is a finite set (terminals) such that $N \cap \Delta = \emptyset$,
- $Z \in N_1$ (initial nonterminal), and
- R is a ranked alphabet (rules) of the form $A(x_1^{(1)}x_1^{(3)}, x_1^{(2)}x_2^{(1)}) \to A_1(x_1^{(1)}, x_2^{(1)})A_2(x_2^{(1)})A_3(x_1^{(3)})$ (rank 3)

Equivalence

RCG $G = (N, \Delta, Z, R)$

$$A(w_1, \ldots, w_n) \to A_1(x_1^{(1)}, \ldots, x_{l_1}^{(1)}) \ldots A_k(x_k^{(1)}, \ldots, x_{l_k}^{(1)})$$

$$A \to \langle w_1, \ldots, w_n \rangle (A_1, \ldots, A_k) \text{ with } \langle w_1, \ldots, w_k \rangle \in \Sigma_{(l_1 \ldots l_k, n)}$$

LCFRS $G' = (N, \Sigma, Z, R)$ over Δ

G and G' are *related*.

Weighted LCFRS

Definition (Weighted LCFRS)

Let $(S, \oplus, 0, \Omega)$ be an M-monoid and $G = (N, \Sigma, Z, R)$ be an LCFRS. A *weighted LCFRS* is a tuple (G, wt) where $\mathsf{wt} : R \to \Omega$ is a rank-preserving mapping.

Example

$$\left(\mathrm{S} \to \langle x_1^{(1)} x_2^{(2)} \rangle(\mathrm{NP}\ \mathrm{VP})\right) \mapsto \left((s_1, s_2) \mapsto 1 \cdot s_1 \cdot s_2\right)$$
$$\left(\mathrm{NP} \to \langle x_1^{(1)} \rangle(\mathrm{NN})\right) \mapsto \left((s) \mapsto 0.7 \cdot s\right)$$
$$\left(\mathrm{NN} \to \langle \mathrm{Local} \rangle\right) \mapsto \left(() \mapsto 0.25\right)$$

M-monoid parsing problem

M-monoid parsing problem

Given

1. a complete M-monoid $(S, \sum^{\oplus})$ with $(S, \oplus, 0, \Omega)$,
2. a weighted LCFRS (G, wt) over S where
   $G = (N, \Sigma, Z, R)$ is an LCFRS over Δ and $\mathrm{wt} : R \to \Omega$, and
3. a sentence $e = e_1 \dots e_n$ with $n \geq 1$ and $e_i \in \Delta$

Compute $\mathrm{parse}(e) = \sum^{\oplus}_{d \in (T_R)_Z : [\![\pi_\Sigma(d)]\!] = e} h(d)$, where

- $h : T_R \to S$ such that $h(d) = g(\widehat{\mathrm{wt}}^1(d))$,
- g is the initial homomorphism $T_\Omega \to S$

[1] deterministic tree relabeling induced by wt

Weighted deductive parsing [Ned03]

Range vector vector $\begin{pmatrix} (l_1, r_1) \\ \dots \\ (l_k, r_k) \end{pmatrix}$ such that $0 \leq l_i < r_i \leq |e|$

Items $\mathcal{I} = \{[A, \vec{\kappa}] \mid A \in N \wedge \vec{\kappa} \in \text{ranges}(e)\}$

Inference rules

SCAN: $\frac{}{[A,(i-1,i)]}$ if $\rho = (A \to \langle e_i \rangle)$ in R

RULE: $\frac{[B_1,\vec{\kappa}_1] \quad \dots \quad [B_k,\vec{\kappa}_k]}{[A,\sigma(\vec{\kappa}_1,\dots,\vec{\kappa}_k)]}$ if $\rho = (A \to \sigma(B_1, \dots, B_k))$ in R

Goal: $[Z, (0, |e|)]$

M-monoid parsing algorithm

Input

1. an M-monoid $(S, \oplus, 0, \Omega)$,
2. an LCFRS$^-$ $G = (N, \Sigma, Z, R)$ over Δ, and $\mathrm{wt} : R \to \Omega$,
3. a function select : $2^{\mathcal{I}} \to \mathcal{I}$, and
4. a sentence $e = e_1 \ldots e_n$ with $n \geq 1$ and $e_i \in \Delta$

Variables $V : \mathcal{I} \to S$ mapping

Output $\mathrm{parse}(e)$

Algorithm 3.1 M-monoid parsing for LCFRS$^-$

$\mathcal{A}, \mathcal{C} \leftarrow \varnothing$

for each $A \in N$ and $\vec{\kappa}$ range vector over e **do**

  $V([A, \vec{\kappa}]) \leftarrow 0$

for each $\rho = (A \to \sigma)$ in R and $[A, \vec{\kappa}]$ generated by SCAN $\frac{}{[A,\vec{\kappa}]}$ **do**

  $V([A, \vec{\kappa}]) \leftarrow V([A, \vec{\kappa}]) \oplus \mathsf{wt}(\rho)()$

  $\mathcal{A} \leftarrow \mathcal{A} \cup \{[A, \vec{\kappa}]\}$

while $\mathcal{A} \neq \emptyset$ **do**

  $[A, \vec{\kappa}] \leftarrow \mathrm{select}(\mathcal{A})$

  $\mathcal{A} \leftarrow \mathcal{A} \setminus \{[A, \vec{\kappa}]\}$

  $\mathcal{C} \leftarrow \mathcal{C} \cup \{[A, \vec{\kappa}]\}$

  for each $\rho = (B \to \sigma(B_1, \ldots B_k))$ in R and $[B, \vec{\eta}]$ deduced by RULE $\frac{*}{[B,\vec{\eta}]}$ from $[A, \vec{\kappa}]$ and other items from $\mathcal{C}$ **do**

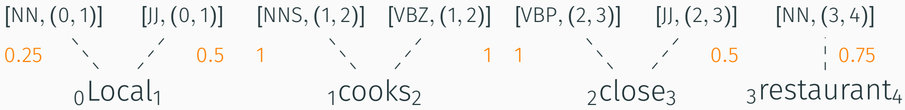
    $V([B, \vec{\eta}]) \leftarrow V([B, \vec{\eta}]) \oplus \mathsf{wt}(\rho)(V([B_1, \vec{\kappa}_1]), \ldots, V([B_k, \vec{\kappa}_k]))$

    if $[B, \vec{\eta}] \notin \mathcal{C}$ **then**

      $\mathcal{A} \leftarrow \mathcal{A} \cup \{[B, \vec{\eta}]\}$

return $V([Z, (0, n)])$

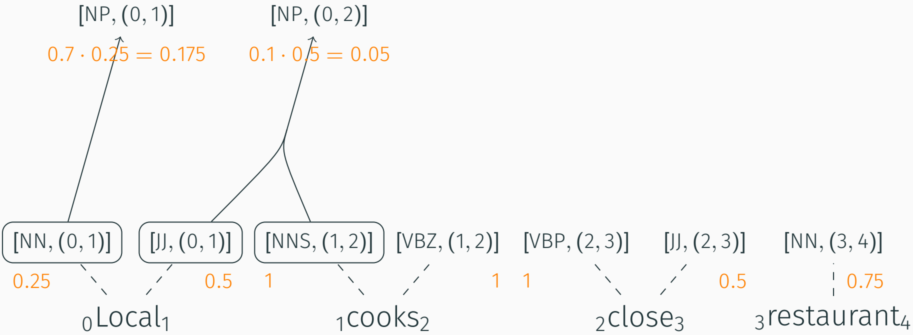
Example run of the algorithm



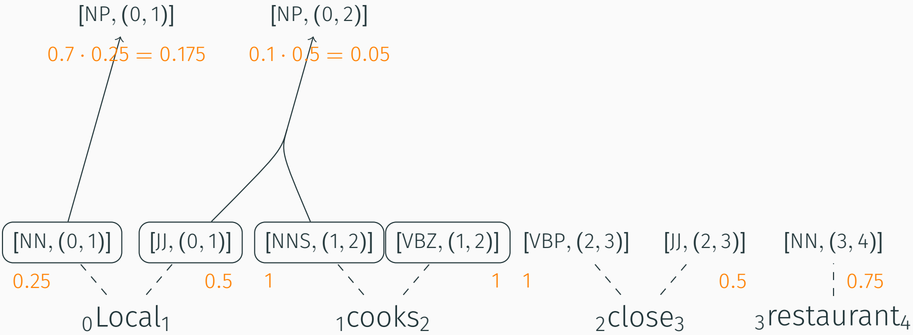
Example run of the algorithm

Example run of the algorithm

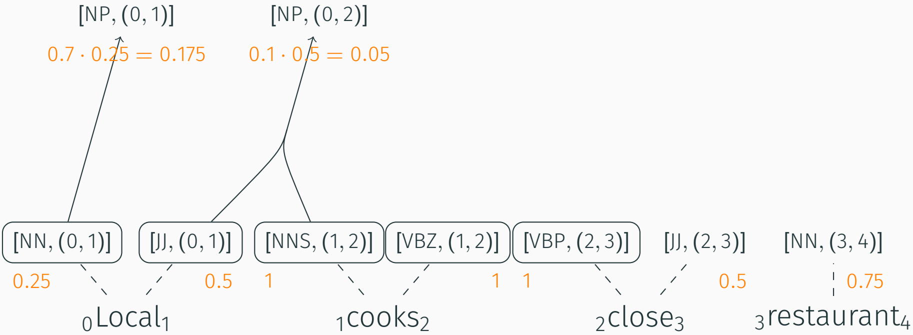
Example run of the algorithm



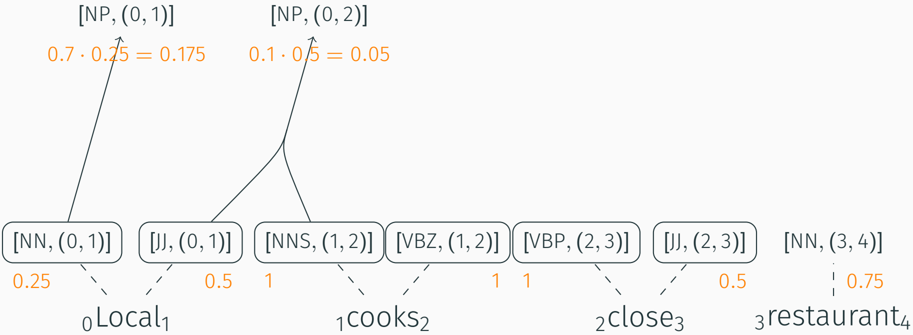
Example run of the algorithm



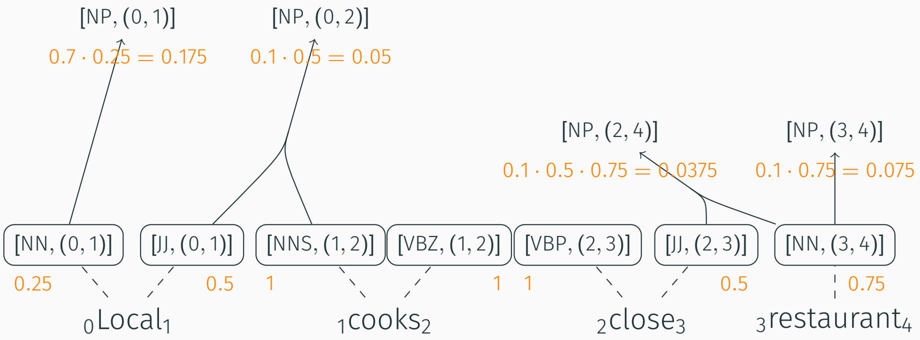
Example run of the algorithm



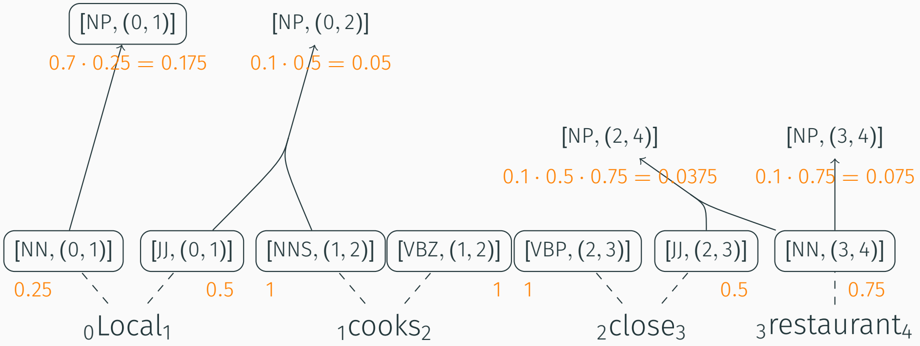
Example run of the algorithm



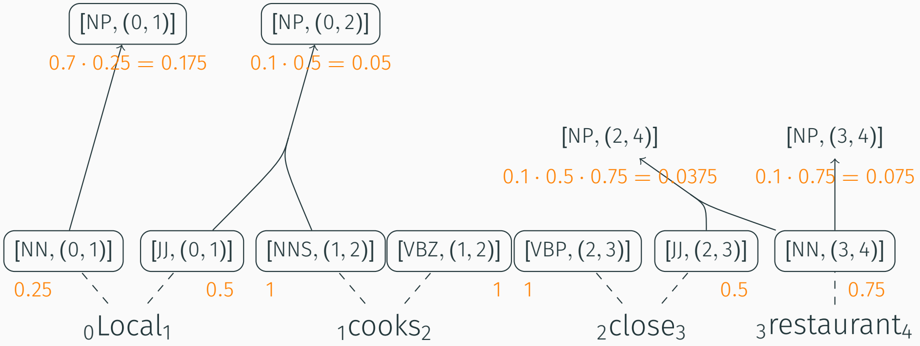
Example run of the algorithm



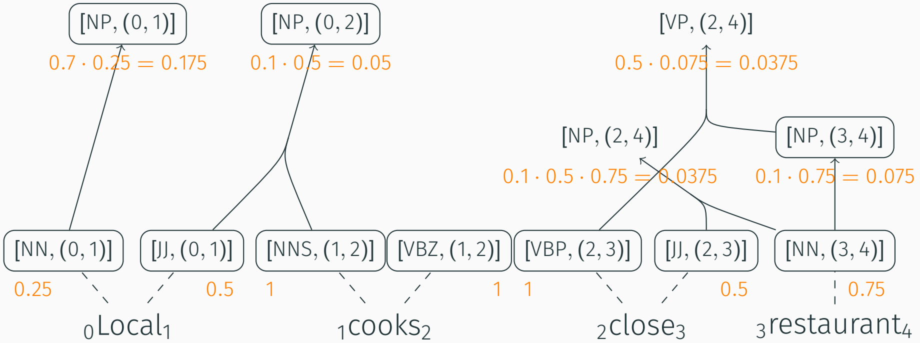
Example run of the algorithm



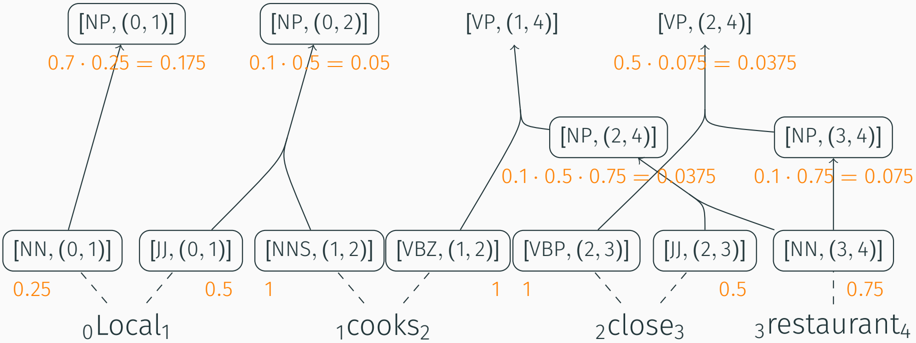
Example run of the algorithm



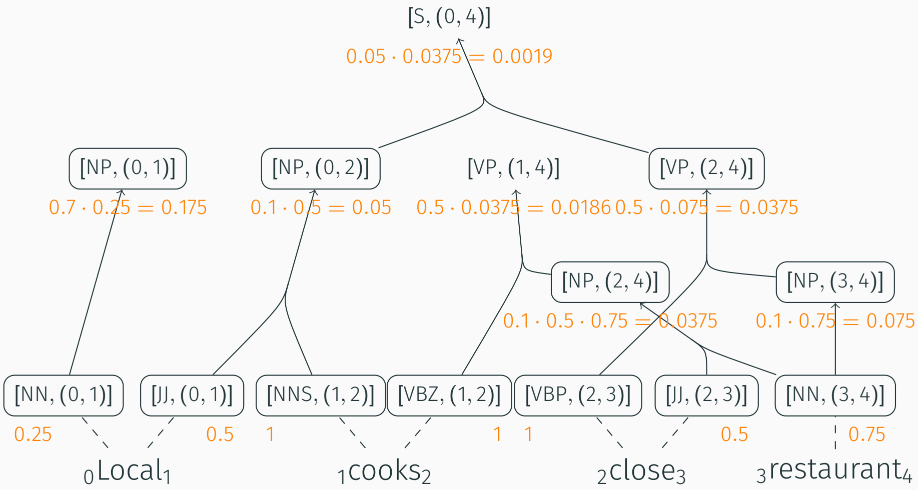
Example run of the algorithm



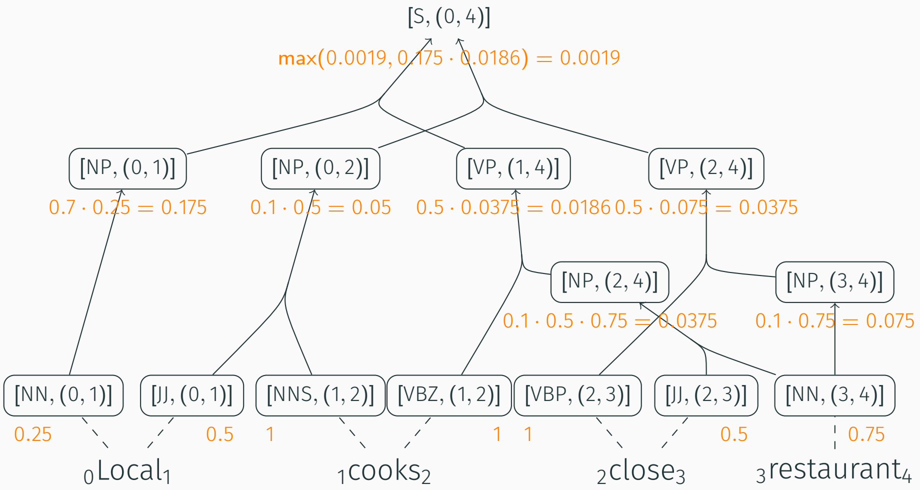
Example run of the algorithm



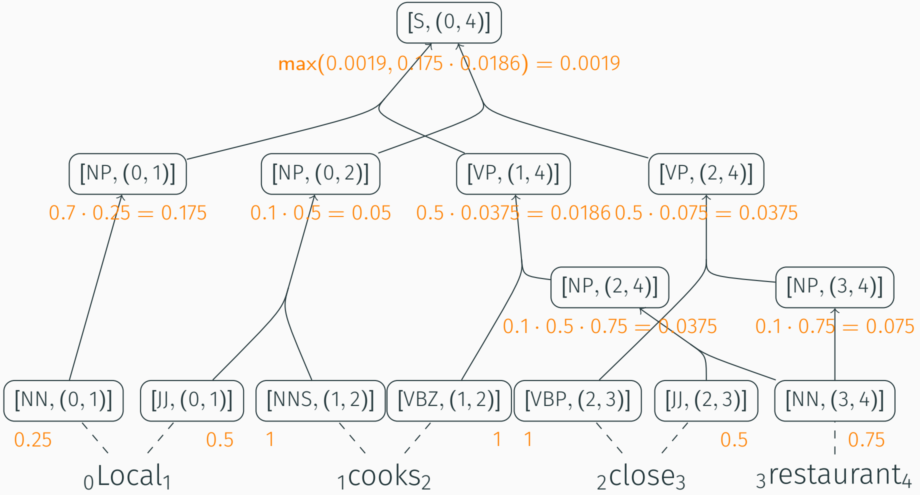
Example run of the algorithm



Example run of the algorithm



Example run of the algorithm



Properties of the algorithm

Termination I

Theorem

The M-monoid parsing algorithm always terminates.

Relevant variables $\mathcal{A}_i$, $\mathcal{C}_i$ (agenda and chart in ith iteration)

$$\mathcal{C}_i \xrightarrow{[A, \vec{\kappa}] \leftarrow \text{select}(\mathcal{A}),\, \mathcal{C} \leftarrow \mathcal{C} \cup \{[A, \vec{\kappa}]\}} \mathcal{C}_{i+1}$$

$\Longrightarrow \mathcal{C}_{i+1} = f(\mathcal{C}_i)$ with $f(\mathcal{C}_i) = \mathcal{C}_i \cup \{[A, \vec{\kappa}]\}$

Termination II

Interpretation $\mathsf{int} : 2^{\mathcal{I}} \rightarrow \mathbb{N}$ with $\mathsf{int}(\mathcal{C}) = n_0 - |\mathcal{C}|$
(n_0 number of all items)

$$\begin{aligned} \mathsf{int}(\mathcal{C}) &= n_0 - |\mathcal{C}| \\ &> n_0 - (|\mathcal{C}| + 1) \\ &= n_0 - |\mathcal{C} \cup \{[A, \vec{\kappa}]\}|^2 \\ &= n_0 - |f(\mathcal{C})| \\ &= \mathsf{int}(f(\mathcal{C})) \end{aligned}$$

[2] $\mathcal{A}$ and $\mathcal{C}$ are always disjoint

Cyclic and acyclic LCFRS

Let $G = (N, \Delta, Z, R)$ be an RCG and $e \in \Delta^+$.

G is cyclic for e (otherwise *acyclic*)

$$Z(e) \Rightarrow_G^* \alpha A(\vec{\kappa}(e))\beta \Rightarrow_G^+ \alpha' A(\vec{\kappa}(e))\beta' \Rightarrow_G^* \varepsilon$$

G is weakly cyclic for e (otherise *weakly acyclic*)

$$A(\vec{\kappa}(e)) \Rightarrow_G^+ \alpha A(\vec{\kappa}(e))\beta \Rightarrow_G^* \varepsilon$$

G is *(weakly) cyclic*
$\Leftrightarrow$ there is an $e \in \Delta^+$ such that G is (weakly) cyclic for e.

Correctness for acyclic LCFRS

Theorem

Let $G = (N, \Sigma, Z, R)$ *be an* LCFRS^- *over* Δ *and* $e \in \Delta^+$ *such that* G *is weakly acyclic for* e*. Then for every* M*-monoid* $(S, \oplus, 0, \Omega)$ *and* $\mathsf{wt} : R \to \Omega$ *there exists a function* $\mathsf{select} : 2^{\mathcal{I}} \to \mathcal{I}$ *such that after termination of Algorithm 3.1 for each* $[A, \vec{\kappa}] \in \mathcal{C}$ *it holds that*

$$V([A, \vec{\kappa}]) = \sum^{\bigoplus}_{d \in (T_R)_A : [\![\pi_\Sigma(d)]\!] = \vec{\kappa}(e)} h(d) \ .$$

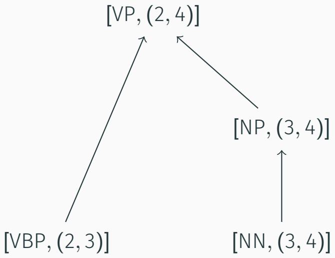
Corollary

Algorithm 3.1 is correct for the class of all acyclic LCFRS^-*.*

Item dependeny graph

$I(G, e) = (V, \leftarrow)$

- V: all items that are useful for e
- $[B, \vec{\eta}] \leftarrow [A, \vec{\kappa}]$ if $[B, \vec{\eta}]$ is derivable from $[A, \vec{\kappa}]$

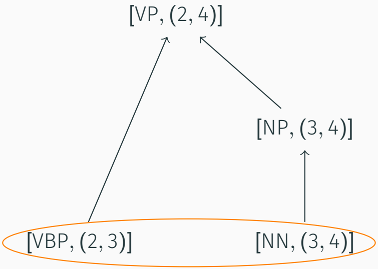


Front $F_{I(G,e)}(\mathcal{C})$ candidates for select

Item dependeny graph

$I(G, e) = (V, \leftarrow)$

- V: all items that are useful for e
- $[B, \vec{\eta}] \leftarrow [A, \vec{\kappa}]$ if $[B, \vec{\eta}]$ is derivable from $[A, \vec{\kappa}]$

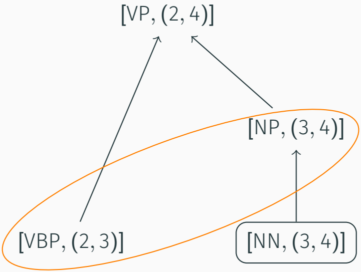


Front $F_{I(G,e)}(\mathcal{C})$ candidates for select

Item dependeny graph

$I(G, e) = (V, \leftarrow)$

- V: all items that are useful for e
- $[B, \vec{\eta}] \leftarrow [A, \vec{\kappa}]$ if $[B, \vec{\eta}]$ is derivable from $[A, \vec{\kappa}]$

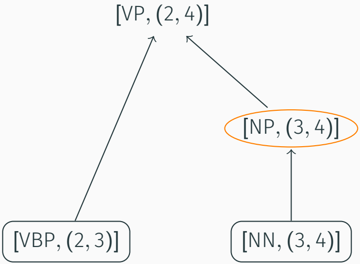


Front $F_{I(G,e)}(\mathcal{C})$ candidates for select

Item dependeny graph

$I(G, e) = (V, \leftarrow)$

- V: all items that are useful for e
- $[B, \vec{\eta}] \leftarrow [A, \vec{\kappa}]$ if $[B, \vec{\eta}]$ is derivable from $[A, \vec{\kappa}]$

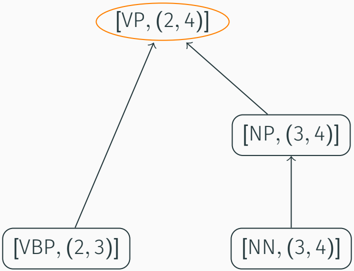


Front $F_{I(G,e)}(\mathcal{C})$ candidates for select

Item dependeny graph

$I(G, e) = (V, \leftarrow)$

- V: all items that are useful for e
- $[B, \vec{\eta}] \leftarrow [A, \vec{\kappa}]$ if $[B, \vec{\eta}]$ is derivable from $[A, \vec{\kappa}]$

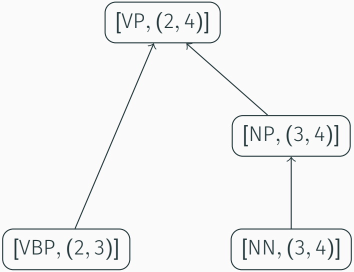


Front $F_{I(G,e)}(\mathcal{C})$ candidates for select

Item dependeny graph

$I(G, e) = (V, \leftarrow)$

- V: all items that are useful for e
- $[B, \vec{\eta}] \leftarrow [A, \vec{\kappa}]$ if $[B, \vec{\eta}]$ is derivable from $[A, \vec{\kappa}]$



Front $F_{I(G,e)}(\mathcal{C})$ candidates for select

Select function

$$\mathsf{select}_{I(G,e)} : 2^{\mathcal{I}} \to \mathcal{I}$$
$$J \mapsto \begin{cases} \text{arb. } j \in F_{I(G,e)}(\mathcal{C}_n) \cap J & \text{if } F_{I(G,e)}(\mathcal{C}_n) \cap J \neq \emptyset \\ \text{arb. } j \in J & \text{otherwise} \end{cases}$$

Proof sketch

Lemma

If G is weakly acyclic for e then for every $n \in \mathbb{N}$ and $[A, \vec{\kappa}] \in V$ it holds that $[A, \vec{\kappa}] \in \mathcal{C}_n$ implies

$$V([A, \vec{\kappa}]) = \sum^{\bigoplus}_{d \in (T_R)_A : [\![\pi_\Sigma(d)]\!] = \vec{\kappa}(e)} h(d)$$

for the nth and all following iterations of the body of the while loop, where $I(G, e) = (V, \leftarrow)$.

Proof.

Follows from the Lemma with $\mathsf{select}_{I(G,e)}$ as the select function. □

Inferior M-monoids i

Superior functions [Knu77, Jun06]

Definition (Inferior operation)

Let $(S, \preceq)$ be a totally ordered set and $\omega : S^k \to S$ ($k \in \mathbb{N}$) be an operation. We call ω $\preceq$-*inferior* if for every $s_1, \ldots, s_k, s \in S$ and for every $i \in \{1, \ldots, k\}$ the following properties hold:

1. $s \preceq s_i \Rightarrow \omega(\ldots, s_{i-1}, s, s_{i+1}, \ldots) \preceq \omega(\ldots, s_{i-1}, s_i, s_{i+1}, \ldots)$
2. $\omega(s_1, \ldots, s_k) \preceq \min\{s_1, \ldots, s_k\}$

Definition (Inferior M-monoid)

Let $(S, \oplus, 0, \Omega)$ be an M-monoid. Moreover, let $\preceq_\oplus$ be the binary relation on S defined for every $a, b \in S$ as follows: $a \preceq_\oplus b$ if $a \oplus b = b$. If $\preceq_\oplus$ is a total order and every $\omega \in \Omega$ is $\preceq_\oplus$-inferior, then we call the M-monoid S *inferior*.

Example: *Viterbi* M-monoid

Correctness for inferior M-monoids

Theorem

Let $\mathcal{S}$ be the class of inferior M-monoids. Algorithm 3.1 is correct for $\mathcal{S}$ and the select function

$$\begin{aligned}\mathrm{select} : 2^{\mathcal{I}} &\to \mathcal{I}\\ J &\mapsto \textit{an arbitrary } j \in \operatorname*{arg\,max}_{j \in J}{}_{\preceq_\oplus} V(j)\end{aligned} \quad ,$$

where

$$\operatorname*{arg\,max}_{j \in J}{}_{\preceq_\oplus} V(j) = \{ j \in J \mid V(j') \preceq_\oplus V(j) \textit{ for every } j' \in j \} \quad .$$

$\mathcal{A}$ and $\mathcal{C}$ are always disjoint

For every $i \in \mathbb{N}$, $\mathcal{A}_i \cap \mathcal{C}_i = \emptyset$.

Induction base clear, as $\mathcal{C}_0 = \emptyset$

Induction step

$$
\begin{aligned}
\mathcal{A}_i \cap \mathcal{C}_i &= ((\mathcal{A}_i \setminus \{[A, \vec{\kappa}]\}) \cup \{i \in \mathcal{I} \mid \cdots \wedge i \notin \mathcal{C}_{i+1}\}) \cap \mathcal{C}_{i+1} \\
&= \mathcal{A}_i \setminus \{[A, \vec{\kappa}]\} \cap (\mathcal{C}_i \cup \{[A, \vec{\kappa}]\}) \\
&\quad \cup (\{i \in \mathcal{I} \mid \cdots \wedge i \notin \mathcal{C}_{i+1}\}) \cap (\mathcal{C}_i \cup \{[A, \vec{\kappa}]\})) \\
&= (\mathcal{A}_i \setminus \{[A, \vec{\kappa}]\} \cap \mathcal{C}_i) \cup (\mathcal{A}_i \setminus \{[A, \vec{\kappa}]\} \cap \{[A, \vec{\kappa}]\}) \cup \emptyset \\
&= \emptyset \cup \emptyset
\end{aligned}
$$

Proof sketch i

Lemma

If G is weakly acyclic for e, then $I(G, e)$ is acyclic.

Lemma

If G is weakly acyclic for e, then $F_{I(G,e)}(\mathcal{C}) \subseteq \mathcal{A}$ is a loop invariant in each iteration of the body of the while loop (lines 7–14).

Proof sketch ii

Corollary

If G is weakly acyclic for e then for every $n \in \mathbb{N}$ and $v, w \in V$ such that $v \leftarrow w$ it holds that $w \in F_{I(G,e)}(\mathcal{C}_n)$ implies $v = \mathsf{select}_{I(G,e)}(\mathcal{A}_i)$ for some $i < n$, where $I(G, e) = (V, \leftarrow)$.

Lemma

If G is weakly acyclic for e, then for every $n \in \mathbb{N}$ and $[A, \vec{\kappa}] \in V$ it holds that $[A, \vec{\kappa}] = \mathsf{select}_{I(G,e)}(A_n)$ implies $[A, \vec{\kappa}] \in F_{I(G,e)}(\mathcal{C}_n)$, where $I(G, e) = (V, \leftarrow)$.

Proof sketch iii

Corollary

If G is weakly acyclic for e, then for every $n \in \mathbb{N}$ *and* $[A, \vec{\kappa}] \in V$ *it holds that* $[A, \vec{\kappa}] \in \mathcal{C}_n$ *implies* $[A, \vec{\kappa}] \in F_{I(G,e)}(\mathcal{C}_i)$ *for some* $i < n$, *where* $I(G, e) = (V, \leftarrow)$.

Proof sketch i

Lemma

For every item $[A, \vec{\kappa}]$ *it holds that once* $[A, \vec{\kappa}]$ *has been added to* $\mathcal{C}$, $V([A, \vec{\kappa}])$ *will not change anymore.*

Lemma

For every item $[A, \vec{\kappa}]$ *it holds that at every point of the computation* $V([A, \vec{\kappa}]) = 0$ *or* $V([A, \vec{\kappa}]) = h(d)$ *for* $d \in (T_R)_A$ *such that* $[\![\pi_\Sigma(d)]\!] = \vec{\kappa}(e)$.

Proof sketch ii

$$W([A,\vec{\kappa}]) = \sum^{\bigoplus}_{d \in (T_R)_A : [\![\pi_\Sigma(d)]\!] = \vec{\kappa}(e)} h(d)$$

Lemma

For every item $[A,\vec{\kappa}]$ *and for every* $d \in (T_R)_A$ *such that* $[\![\pi_\Sigma(d)]\!] = \vec{\kappa}(e)$ *it holds that* $h(d) \preceq_\oplus W([A,\vec{\kappa}])$.

Lemma

For every item $[A,\vec{\kappa}]$ *it holds that at every point of the computation* $V([A,\vec{\kappa}]) \preceq_\oplus W([A,\vec{\kappa}])$.

Proof sketch iii

Proof.

$$[A, \vec{\kappa}] \in \mathcal{C} \rightarrow V([A, \vec{\kappa}]) = W([A, \vec{\kappa}]) \tag{I}$$

is a loop invariant for every $A \in N$ and $\vec{\kappa} \in \mathsf{ranges}(e)$.

Proof by contradiction similar to [Knu77, Jun06]. □

Jean Christoph Jung.
Knuth's generalization of Dijkstra's algorithm.
2006.

D.E. Knuth.
A Generalization of Dijkstra's Algorithm.
Inform. Process. Lett., 6(1):1–5, February 1977.

M.-J. Nederhof.
Weighted deductive parsing and Knuth's algorithm.
Computational Linguistics, 29(1):135–143, 2003.