Training of hybrid grammars for the generation of discontinuous phrase structures and non-projective dependency structures

Diplomverteidigung

Kilian Gebhardt

Professur Grundlagen der Programmierung
Institut für Theoretische Informatik
Technische Universität Dresden

16. September 2015

syntactic structures

What shall I do ?

What shall I do ?

syntactic structures

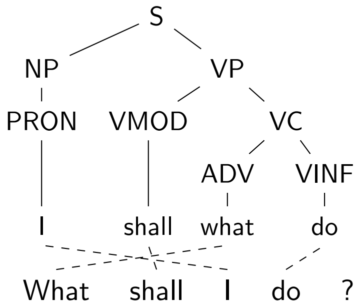
phrase structures

What shall I do ?

What shall I do ?

syntactic structures

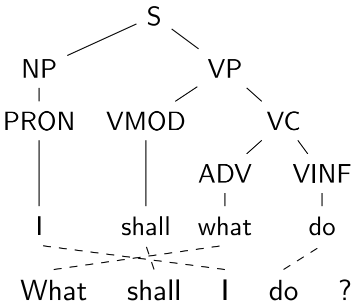
phrase structures



What shall I do ?

syntactic structures

phrase structures

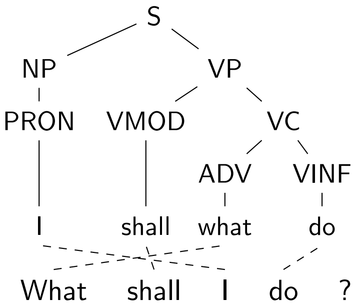


dependency structures

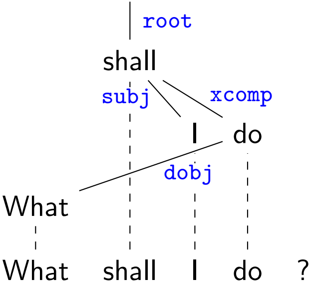
What shall I do ?

syntactic structures

phrase structures

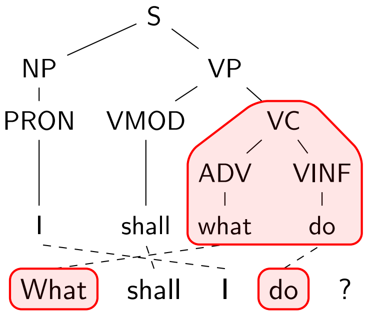


dependency structures



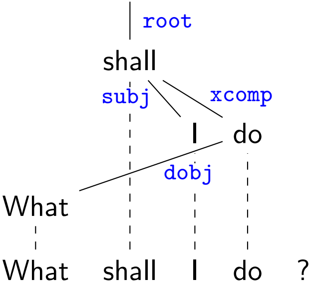
syntactic structures

phrase structures



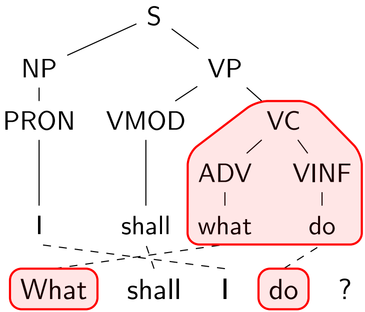
discontinuous

dependency structures



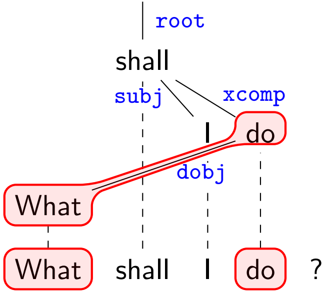
syntactic structures

phrase structures



discontinuous

dependency structures



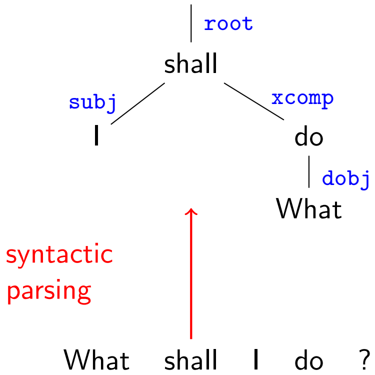
non-projective

objective

a formal model for

objective

a formal model for

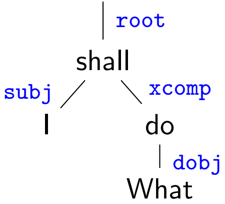


1. hybrid trees and (LCFRS,sDCP)-hybrid grammars

2. grammar induction

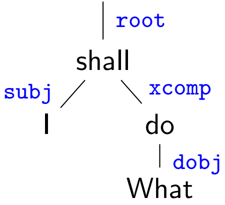
3. experiments

hybrid trees and hybrid grammars [NV14]



is linearized to ‘What shall I do?’

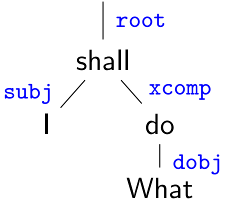
hybrid trees and hybrid grammars [NV14]



is linearized to 'What shall I do?'

implicit ✗

hybrid trees and hybrid grammars [NV14]



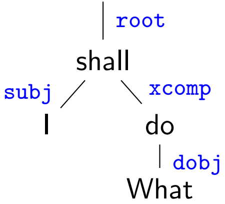
is linearized to ‘What shall I do?’

implicit ✗

explicit ✓

hybrid tree

hybrid trees and hybrid grammars [NV14]



is linearized to 'What shall I do?'

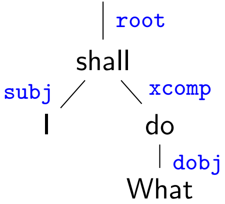
implicit ✗

explicit ✓

hybrid tree

= tree

hybrid trees and hybrid grammars [NV14]



What shall I do ?

is linearized to ‘What shall I do?’

implicit ✗

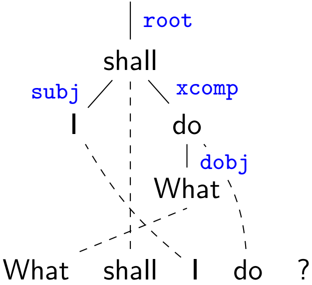
explicit ✓

hybrid tree

= tree

+ string

hybrid trees and hybrid grammars [NV14]



is linearized to ‘What shall I do?’

implicit ✗

explicit ✓

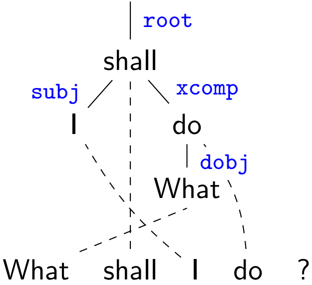
hybrid tree

= tree

+ string

+ sync.

hybrid trees and hybrid grammars [NV14]



is linearized to 'What shall I do?'

implicit ✗

explicit ✓

hybrid tree

= tree

+ string

+ sync.

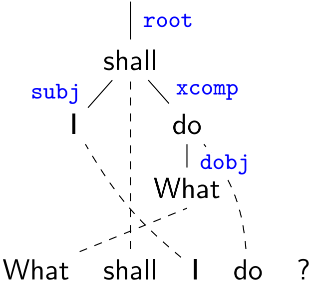
hybrid grammar = string grammar + tree grammar

- synchronize derivational nonterminals
  and synchronize terminals

our choice:

- string grammar: linear context-free rewriting system (LCFRS)
- tree grammar: simple definite clause program (sDCP)

hybrid trees and hybrid grammars [NV14]



is linearized to ‘What shall I do?’

implicit ✗

explicit ✓

hybrid tree

= tree

+ string

+ sync.

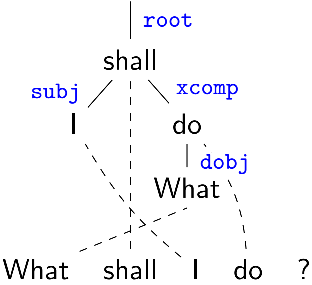
hybrid grammar = string grammar + tree grammar

- synchronize derivational nonterminals
  and synchronize terminals

our choice:

- string grammar: linear context-free rewriting system (LCFRS)
- tree grammar: simple definite clause program (sDCP)

hybrid trees and hybrid grammars [NV14]



is linearized to ‘What shall I do?’

implicit ✗

explicit ✓

hybrid tree

= tree

+ string

+ sync.

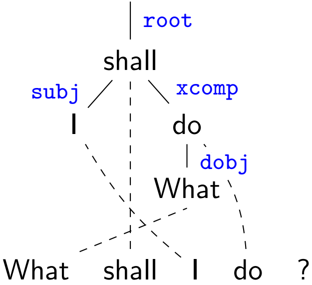
hybrid grammar = string grammar + tree grammar

- synchronize derivational nonterminals
  and synchronize terminals

our choice:

- string grammar: linear context-free rewriting system (LCFRS)
- tree grammar: simple definite clause program (sDCP)

hybrid trees and hybrid grammars [NV14]



is linearized to 'What shall I do?'

implicit ✗

explicit ✓

hybrid tree

= tree

+ string

+ sync.

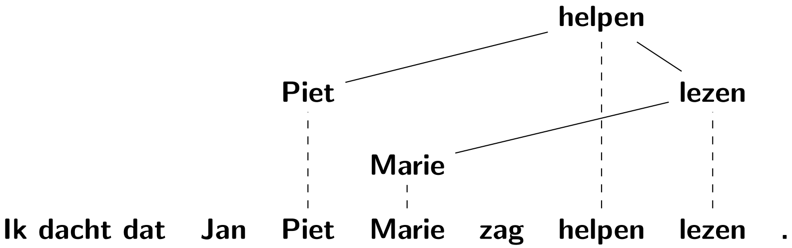
hybrid grammar = string grammar + tree grammar

- synchronize derivational nonterminals
  and synchronize terminals

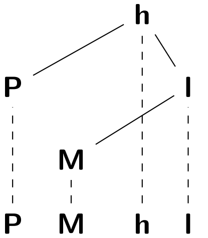
our choice:

- string grammar: linear context-free rewriting system (LCFRS)
- tree grammar: simple definite clause program (sDCP)

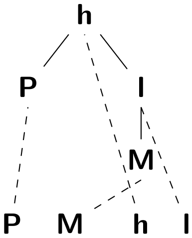
running example



is abbreviated as



or



linear context-free rewriting systems (LCFRSs)

$$
\begin{array}{lcll}
S & \rightarrow & A & C \\
A & \rightarrow & B & \\
B & \rightarrow & \varepsilon & \\
C & \rightarrow & D & \\
D & \rightarrow & \varepsilon &
\end{array}
$$

fanout k, rank r
parsing complexity:
$\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$
\begin{array}{lcll}
S(\mathbf{P}\,\mathbf{M}\,\mathbf{h}\,\mathbf{l}) & \Rightarrow & A(\mathbf{P}, \mathbf{h}) & C(\mathbf{M}, \mathbf{l}) \\
 & \Rightarrow & B(\mathbf{P}) & C(\mathbf{M}, \mathbf{l}) \\
 & \Rightarrow & & C(\mathbf{M}, \mathbf{l}) \\
 & \Rightarrow & & D(\mathbf{M}) \\
 & \Rightarrow & \varepsilon &
\end{array}
$$

$k = 1, r = 2$:
$\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$:
$\mathcal{O}(n^6 \cdot |G|)$

linear context-free rewriting systems (LCFRSs)

$$S \rightarrow A\,(x_1, x_2)\;\; C\,(x_3, x_4)$$
$$A \rightarrow B\,(x_1)$$
$$B \rightarrow \varepsilon$$
$$C \rightarrow D\,(x_1)$$
$$D \rightarrow \varepsilon$$

fanout k, rank r
parsing complexity:
$\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$S(\mathbf{P}\,\mathbf{M}\,\mathbf{h}\,\mathbf{I}) \Rightarrow A(\mathbf{P}, \mathbf{h})\;\; C(\mathbf{M}, \mathbf{I})$$
$$\Rightarrow B(\mathbf{P})\;\; C(\mathbf{M}, \mathbf{I})$$
$$\Rightarrow C(\mathbf{M}, \mathbf{I})$$
$$\Rightarrow D(\mathbf{M})$$
$$\Rightarrow \varepsilon$$

$k = 1, r = 2$:
$\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$:
$\mathcal{O}(n^6 \cdot |G|)$

linear context-free rewriting systems (LCFRSs)

$$
\begin{array}{lcl}
S\,(x_1\,x_3\,x_2\,x_4) & \rightarrow & A\,(x_1, x_2)\;\; C\,(x_3, x_4) \\
A\,(x_1, \mathbf{h}) & \rightarrow & B\,(x_1) \\
B\,(\mathbf{P}) & \rightarrow & \varepsilon \\
C\,(x_1, \mathbf{I}) & \rightarrow & D\,(x_1) \\
D\,(\mathbf{M}) & \rightarrow & \varepsilon
\end{array}
$$

fanout k, rank r
parsing complexity:
$\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$
\begin{array}{lcll}
S(\mathbf{P}\,\mathbf{M}\,\mathbf{h}\,\mathbf{I}) & \Rightarrow & A(\mathbf{P}, \mathbf{h}) & C(\mathbf{M}, \mathbf{I}) \\
 & \Rightarrow & B(\mathbf{P}) & C(\mathbf{M}, \mathbf{I}) \\
 & \Rightarrow & & C(\mathbf{M}, \mathbf{I}) \\
 & \Rightarrow & & D(\mathbf{M}) \\
 & \Rightarrow & \varepsilon &
\end{array}
$$

$k = 1, r = 2$:
$\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$:
$\mathcal{O}(n^6 \cdot |G|)$

linear context-free rewriting systems (LCFRSs)

$$
\begin{array}{lcl}
S\,(x_1\,x_3\,x_2\,x_4) & \rightarrow & A\,(x_1, x_2)\;\; C\,(x_3, x_4) \\
A\,(x_1, \mathbf{h}) & \rightarrow & B\,(x_1) \\
B\,(\mathbf{P}) & \rightarrow & \varepsilon \\
C\,(x_1, \mathbf{l}) & \rightarrow & D\,(x_1) \\
D\,(\mathbf{M}) & \rightarrow & \varepsilon
\end{array}
$$

fanout k, rank r
parsing complexity:
$\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$
\begin{array}{lcll}
S(\mathbf{P}\,\mathbf{M}\,\mathbf{h}\,\mathbf{l}) & \Rightarrow & A(\mathbf{P}, \mathbf{h}) & C(\mathbf{M}, \mathbf{l}) \\
 & \Rightarrow & B(\mathbf{P}) & C(\mathbf{M}, \mathbf{l}) \\
 & \Rightarrow & & C(\mathbf{M}, \mathbf{l}) \\
 & \Rightarrow & & D(\mathbf{M}) \\
 & \Rightarrow & \varepsilon &
\end{array}
$$

$k = 1, r = 2$:
$\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$:
$\mathcal{O}(n^6 \cdot |G|)$

linear context-free rewriting systems (LCFRSs)

$$S(x_1 x_3 x_2 x_4) \rightarrow A(x_1, x_2)\ C(x_3, x_4)$$
$$A(x_1, \mathbf{h}) \rightarrow B(x_1)$$
$$B(\mathbf{P}) \rightarrow \varepsilon$$
$$C(x_1, \mathbf{I}) \rightarrow D(x_1)$$
$$D(\mathbf{M}) \rightarrow \varepsilon$$

fanout k, rank r
parsing complexity:
$\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$S(\mathbf{P}\ \mathbf{M}\ \mathbf{h}\ \mathbf{I}) \Rightarrow A(\mathbf{P}, \mathbf{h})\ C(\mathbf{M}, \mathbf{I}) \Rightarrow B(\mathbf{P})\ C(\mathbf{M}, \mathbf{I}) \Rightarrow C(\mathbf{M}, \mathbf{I}) \Rightarrow D(\mathbf{M}) \Rightarrow \varepsilon$$

$k = 1, r = 2$:
$\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$:
$\mathcal{O}(n^6 \cdot |G|)$

linear context-free rewriting systems (LCFRSs)

- $S\,(x_1\, x_3\, x_2\, x_4) \rightarrow A\,(x_1, x_2)\;\; C\,(x_3, x_4)$
  $A\,(x_1, \mathbf{h}) \rightarrow B\,(x_1)$
  $B\,(\mathbf{P}) \rightarrow \varepsilon$
  $C\,(x_1, \mathbf{l}) \rightarrow D\,(x_1)$
  $D\,(\mathbf{M}) \rightarrow \varepsilon$

fanout k, rank r
parsing complexity:
$\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$S(\mathbf{P}\,\mathbf{M}\,\mathbf{h}\,\mathbf{l}) \Rightarrow A(\mathbf{P}, \mathbf{h})\;\; C(\mathbf{M}, \mathbf{l})$$
$$\Rightarrow B(\mathbf{P})\;\; C(\mathbf{M}, \mathbf{l})$$
$$\Rightarrow C(\mathbf{M}, \mathbf{l})$$
$$\Rightarrow D(\mathbf{M})$$
$$\Rightarrow \varepsilon$$

$k = 1, r = 2$:
$\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$:
$\mathcal{O}(n^6 \cdot |G|)$

linear context-free rewriting systems (LCFRSs)

$$
\begin{array}{lllll}
& S\,(x_1\,x_3\,x_2\,x_4) & \rightarrow & A\,(x_1, x_2) & C\,(x_3, x_4) \\
\blacktriangleright & A\,(x_1, \mathbf{h}) & \rightarrow & B\,(x_1) & \\
& B\,(\mathbf{P}) & \rightarrow & \varepsilon & \\
& C\,(x_1, \mathbf{l}) & \rightarrow & D\,(x_1) & \\
& D\,(\mathbf{M}) & \rightarrow & \varepsilon &
\end{array}
$$

fanout k, rank r
parsing complexity: $\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$
\begin{array}{llll}
S(\mathbf{P\,M\,h\,l}) & \Rightarrow & A(\mathbf{P}, \mathbf{h}) & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & B(\mathbf{P}) & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & & D(\mathbf{M}) \\
& \Rightarrow & \varepsilon &
\end{array}
$$

$k = 1, r = 2$: $\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$: $\mathcal{O}(n^6 \cdot |G|)$

linear context-free rewriting systems (LCFRSs)

$$
\begin{array}{lcl}
S(x_1 x_3 x_2 x_4) & \rightarrow & A(x_1, x_2)\ C(x_3, x_4) \\
A(x_1, \mathbf{h}) & \rightarrow & B(x_1) \\
\blacktriangleright\ B(\mathbf{P}) & \rightarrow & \varepsilon \\
C(x_1, \mathbf{l}) & \rightarrow & D(x_1) \\
D(\mathbf{M}) & \rightarrow & \varepsilon
\end{array}
$$

fanout k, rank r
parsing complexity: $\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$
\begin{array}{lll}
S(\mathbf{P}\ \mathbf{M}\ \mathbf{h}\ \mathbf{l}) & \Rightarrow & A(\mathbf{P}, \mathbf{h})\ \ C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & B(\mathbf{P})\ \ C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & D(\mathbf{M}) \\
& \Rightarrow & \varepsilon
\end{array}
$$

$k = 1, r = 2$: $\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$: $\mathcal{O}(n^6 \cdot |G|)$

linear context-free rewriting systems (LCFRSs)

$$
\begin{array}{llll}
& S\,(x_1\,x_3\,x_2\,x_4) & \rightarrow & A\,(x_1, x_2)\;\; C\,(x_3, x_4) \\
& A\,(x_1, \mathbf{h}) & \rightarrow & B\,(x_1) \\
& B\,(\mathbf{P}) & \rightarrow & \varepsilon \\
\blacktriangleright & C\,(x_1, \mathbf{l}) & \rightarrow & D\,(x_1) \\
& D\,(\mathbf{M}) & \rightarrow & \varepsilon
\end{array}
$$

fanout k, rank r
parsing complexity:
$\mathcal{O}(n^{(r+1)k} \cdot |G|)$

$k = 1, r = 2$:
$\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$:
$\mathcal{O}(n^6 \cdot |G|)$

derivation:

$$
\begin{array}{llll}
S(\mathbf{P}\,\mathbf{M}\,\mathbf{h}\,\mathbf{l}) & \Rightarrow & A(\mathbf{P}, \mathbf{h}) & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & B(\mathbf{P}) & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & & D(\mathbf{M}) \\
& \Rightarrow & \varepsilon &
\end{array}
$$

linear context-free rewriting systems (LCFRSs)

$$
\begin{aligned}
S\,(x_1\, x_3\, x_2\, x_4) &\rightarrow A\,(x_1, x_2)\;\; C\,(x_3, x_4)\\
A\,(x_1, \mathbf{h}) &\rightarrow B\,(x_1)\\
B\,(\mathbf{P}) &\rightarrow \varepsilon\\
C\,(x_1, \mathbf{l}) &\rightarrow D\,(x_1)\\
D\,(\mathbf{M}) &\rightarrow \varepsilon
\end{aligned}
$$

fanout k, rank r
parsing complexity: $\mathcal{O}(n^{(r+1)k} \cdot |G|)$

$k = 1, r = 2$: $\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$: $\mathcal{O}(n^6 \cdot |G|)$

derivation:

$$
\begin{aligned}
S(\mathbf{P}\;\mathbf{M}\;\mathbf{h}\;\mathbf{l}) &\Rightarrow A(\mathbf{P}, \mathbf{h})\;\; C(\mathbf{M}, \mathbf{l})\\
&\Rightarrow B(\mathbf{P})\;\; C(\mathbf{M}, \mathbf{l})\\
&\Rightarrow C(\mathbf{M}, \mathbf{l})\\
&\Rightarrow D(\mathbf{M})\\
&\Rightarrow \varepsilon
\end{aligned}
$$

linear context-free rewriting systems (LCFRSs)

$$
\begin{array}{lcl}
S\ (x_1\, x_3\, x_2\, x_4) & \rightarrow & A\ (x_1, x_2)\quad C\ (x_3, x_4) \\
A\ (x_1, \mathbf{h}) & \rightarrow & B\ (x_1) \\
B\ (\mathbf{P}) & \rightarrow & \varepsilon \\
C\ (x_1, \mathbf{l}) & \rightarrow & D\ (x_1) \\
D\ (\mathbf{M}) & \rightarrow & \varepsilon
\end{array}
$$

fanout k, rank r
parsing complexity: $\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$
\begin{array}{lcll}
S(\mathbf{P}\,\mathbf{M}\,\mathbf{h}\,\mathbf{l}) & \Rightarrow & A(\mathbf{P}, \mathbf{h}) & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & B(\mathbf{P}) & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & & C(\mathbf{M}, \mathbf{l}) \\
& \Rightarrow & & D(\mathbf{M}) \\
& \Rightarrow & \varepsilon &
\end{array}
$$

$k = 1, r = 2$: $\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$: $\mathcal{O}(n^6 \cdot |G|)$

linear context-free rewriting systems (LCFRSs)

$$
\begin{aligned}
S\,(x_1\,x_3\,x_2\,x_4) &\rightarrow A\,(x_1, x_2)\;\; C\,(x_3, x_4)\\
A\,(x_1, \mathbf{h}) &\rightarrow B\,(x_1)\\
B\,(\mathbf{P}) &\rightarrow \varepsilon\\
C\,(x_1, \mathbf{l}) &\rightarrow D\,(x_1)\\
D\,(\mathbf{M}) &\rightarrow \varepsilon
\end{aligned}
$$

fanout k, rank r
parsing complexity:
$\mathcal{O}(n^{(r+1)k} \cdot |G|)$

derivation:

$$
\begin{aligned}
S(\mathbf{P}\,\mathbf{M}\,\mathbf{h}\,\mathbf{l}) &\Rightarrow A(\mathbf{P}, \mathbf{h})\;\; C(\mathbf{M}, \mathbf{l})\\
&\Rightarrow B(\mathbf{P})\;\; C(\mathbf{M}, \mathbf{l})\\
&\Rightarrow C(\mathbf{M}, \mathbf{l})\\
&\Rightarrow D(\mathbf{M})\\
&\Rightarrow \varepsilon
\end{aligned}
$$

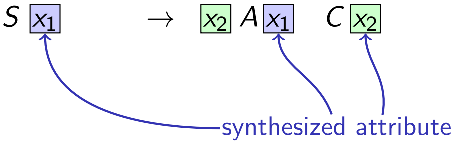
$k = 1, r = 2$:
$\mathcal{O}(n^3 \cdot |G|)$

$k = 2, r = 2$:
$\mathcal{O}(n^6 \cdot |G|)$

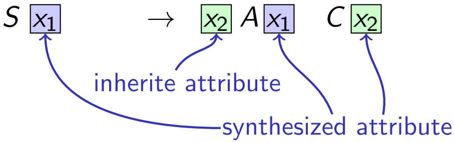
simple definite clause programs (sDCPs)

$$S\ \boxed{x_1} \quad \rightarrow \quad \boxed{x_2}\ A\ \boxed{x_1} \quad C\ \boxed{x_2}$$

simple definite clause programs (sDCPs)



simple definite clause programs (sDCPs)



simple definite clause programs (sDCPs)

$$S\ \boxed{x_1} \rightarrow \boxed{x_2}\ A\ \boxed{x_1}\quad C\ \boxed{x_2}$$

$$\boxed{x_1}\ A\ \boxed{\mathbf{h}(\boxed{x_2}, \boxed{x_1})} \rightarrow B\ \boxed{x_2} \qquad\qquad B\ \boxed{\mathbf{P}} \rightarrow \varepsilon$$

$$C\ \boxed{\mathbf{l}(\boxed{x_1})} \rightarrow D\ \boxed{x_1} \qquad\qquad D\ \boxed{\mathbf{M}} \rightarrow \varepsilon$$

simple definite clause programs (sDCPs)

$$S\ \boxed{x_1} \rightarrow \boxed{x_2}\ A\ \boxed{x_1}\quad C\ \boxed{x_2}$$

$$\boxed{x_1}\ A\ \boxed{\mathbf{h}(\boxed{x_2}, \boxed{x_1})} \rightarrow B\ \boxed{x_2} \qquad\qquad B\ \boxed{\mathbf{P}} \rightarrow \varepsilon$$

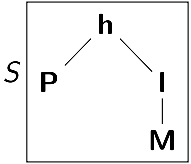
$$C\ \boxed{\mathbf{l}(\boxed{x_1})} \rightarrow D\ \boxed{x_1} \qquad\qquad D\ \boxed{\mathbf{M}} \rightarrow \varepsilon$$

simple definite clause programs (sDCPs)

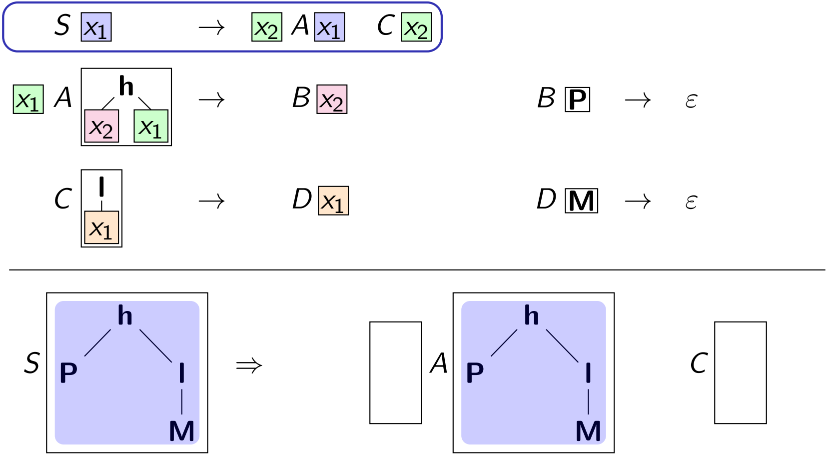
$$S\ \boxed{x_1} \rightarrow \boxed{x_2}\ A\ \boxed{x_1}\quad C\ \boxed{x_2}$$

$$\boxed{x_1}\ A\ \boxed{\mathbf{h}(\boxed{x_2}, \boxed{x_1})} \rightarrow B\ \boxed{x_2} \qquad B\ \boxed{\mathbf{P}} \rightarrow \varepsilon$$

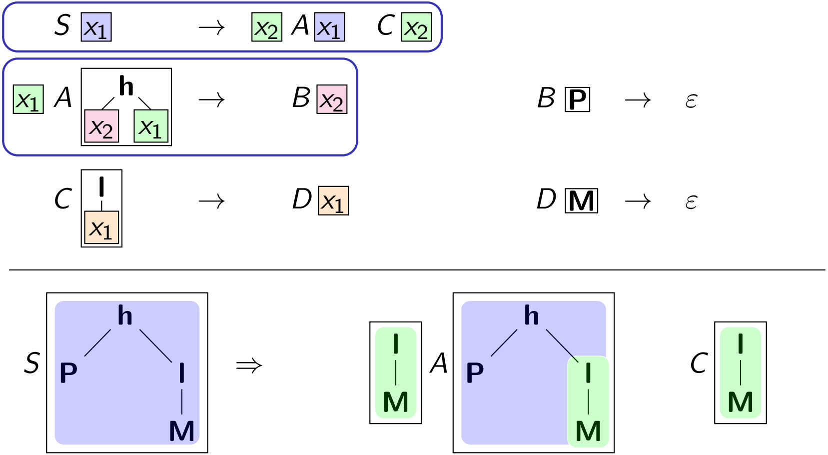
$$C\ \boxed{\mathbf{l}(\boxed{x_1})} \rightarrow D\ \boxed{x_1} \qquad D\ \boxed{\mathbf{M}} \rightarrow \varepsilon$$



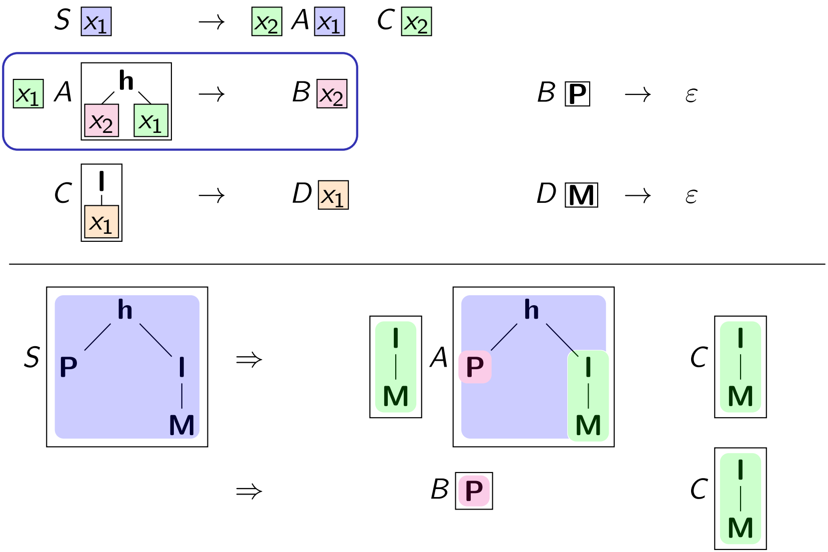
simple definite clause programs (sDCPs)



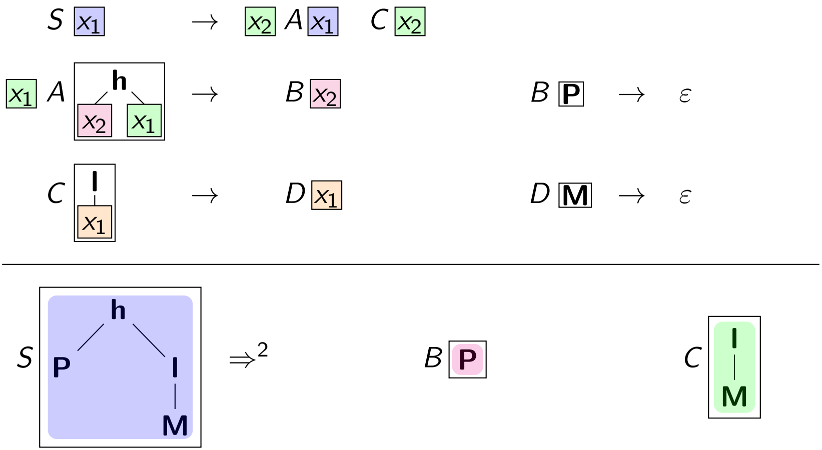
simple definite clause programs (sDCPs)



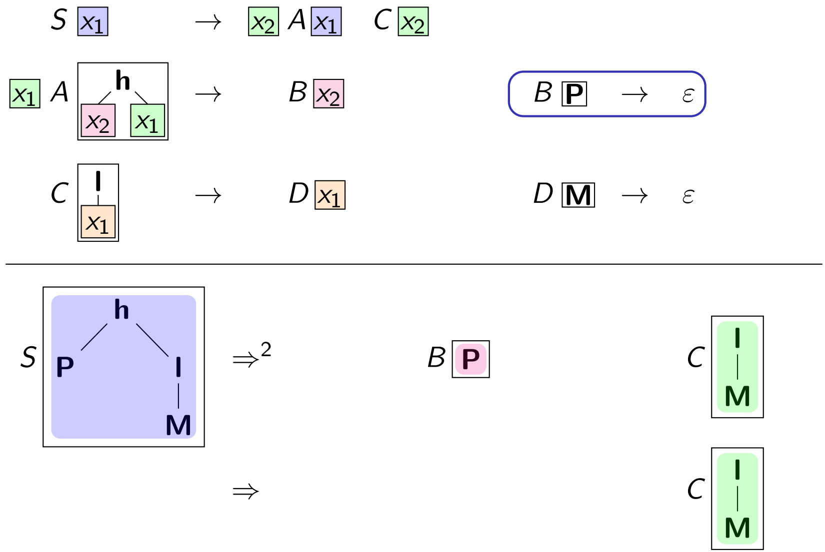
simple definite clause programs (sDCPs)



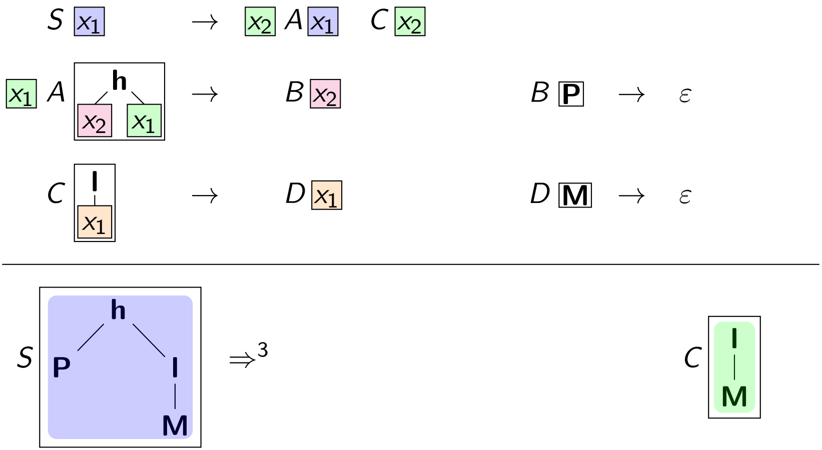
simple definite clause programs (sDCPs)



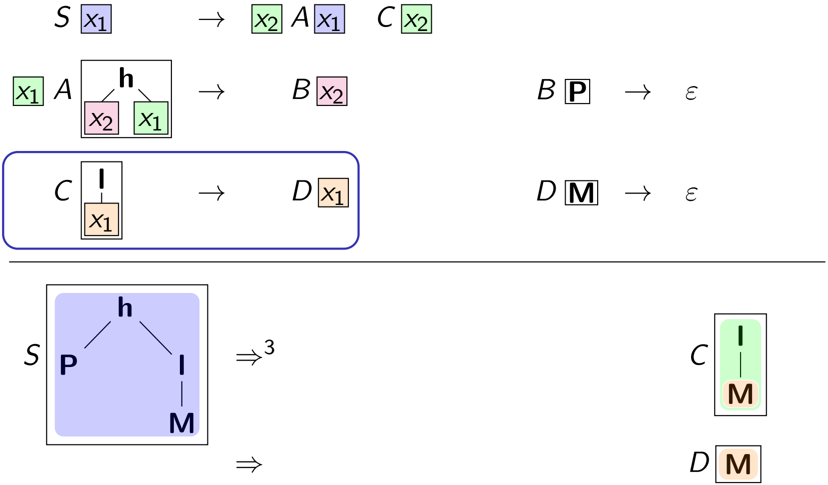
simple definite clause programs (sDCPs)



simple definite clause programs (sDCPs)



simple definite clause programs (sDCPs)



simple definite clause programs (sDCPs)

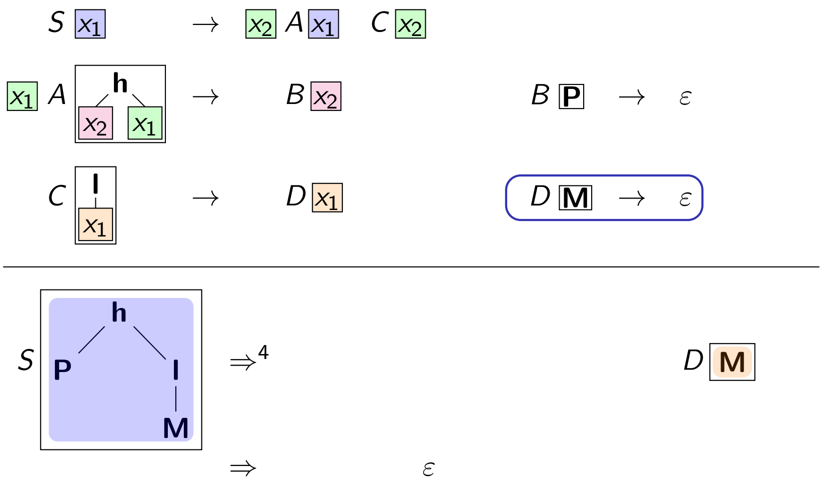
$$S\ \boxed{x_1} \rightarrow \boxed{x_2}\ A\ \boxed{x_1} \quad C\ \boxed{x_2}$$

$$\boxed{x_1}\ A\ \boxed{\mathbf{h}(\boxed{x_2}, \boxed{x_1})} \rightarrow B\ \boxed{x_2} \qquad\qquad B\ \boxed{\mathbf{P}} \rightarrow \varepsilon$$

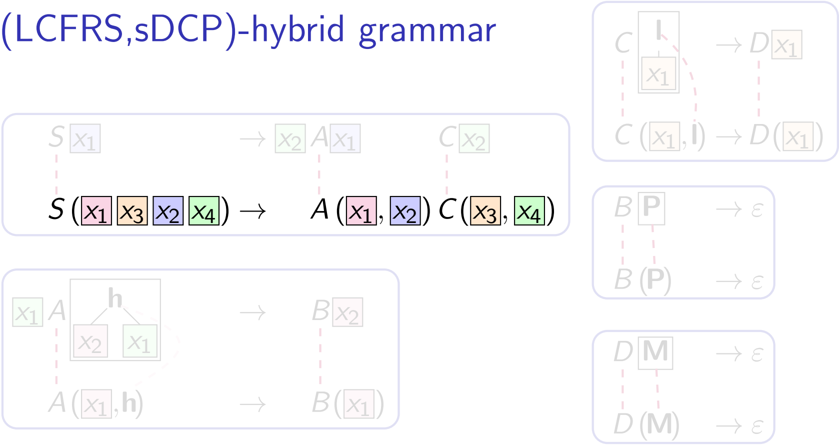
$$C\ \boxed{\mathbf{l}(\boxed{x_1})} \rightarrow D\ \boxed{x_1} \qquad\qquad D\ \boxed{\mathbf{M}} \rightarrow \varepsilon$$

$$S\ \boxed{\mathbf{h}(\mathbf{P}, \mathbf{l}(\mathbf{M}))} \Rightarrow^4 D\ \boxed{\mathbf{M}}$$

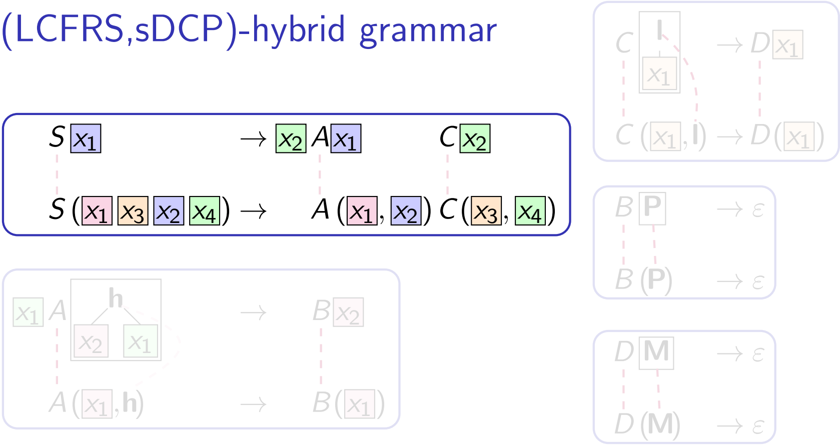
simple definite clause programs (sDCPs)



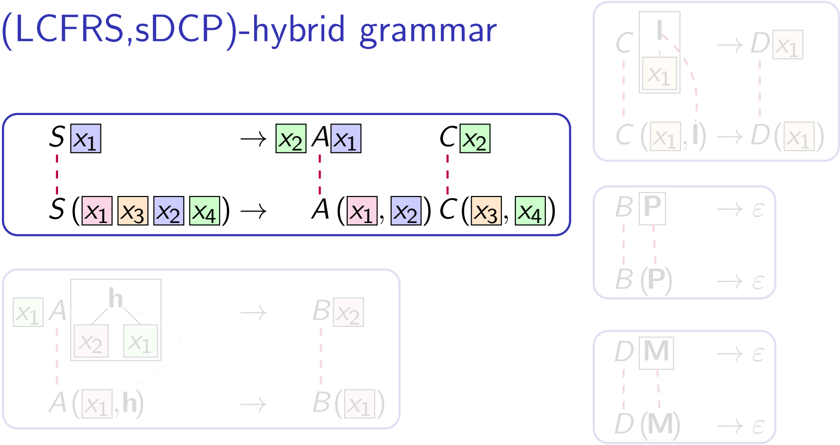
(LCFRS,sDCP)-hybrid grammar



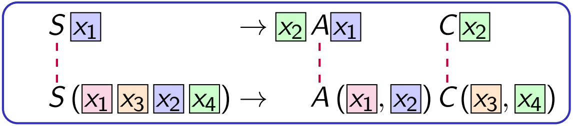
(LCFRS,sDCP)-hybrid grammar

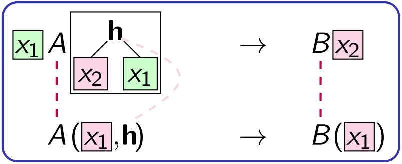


(LCFRS,sDCP)-hybrid grammar

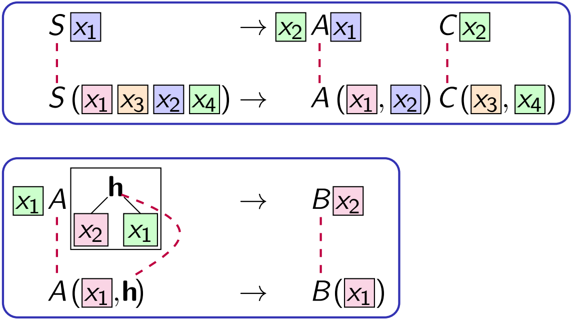


(LCFRS,sDCP)-hybrid grammar

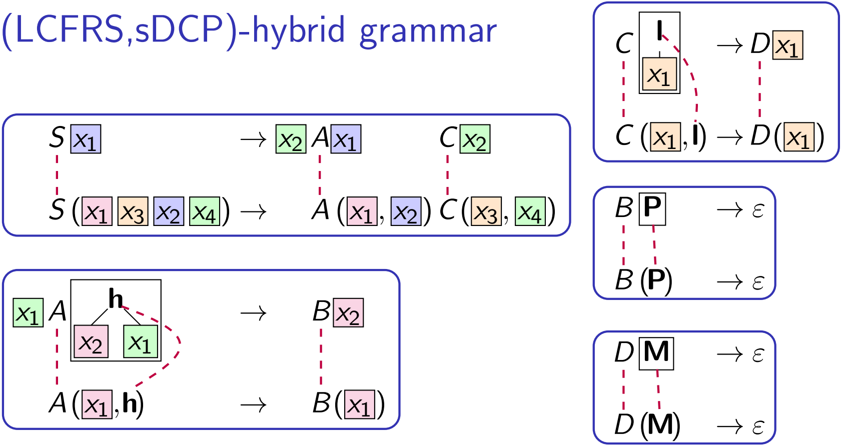




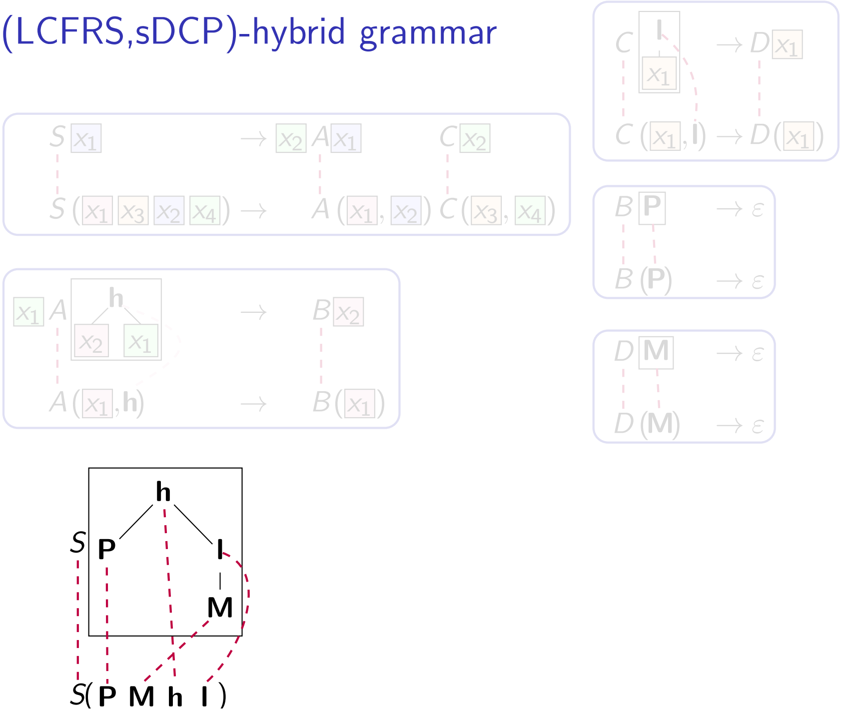
(LCFRS,sDCP)-hybrid grammar



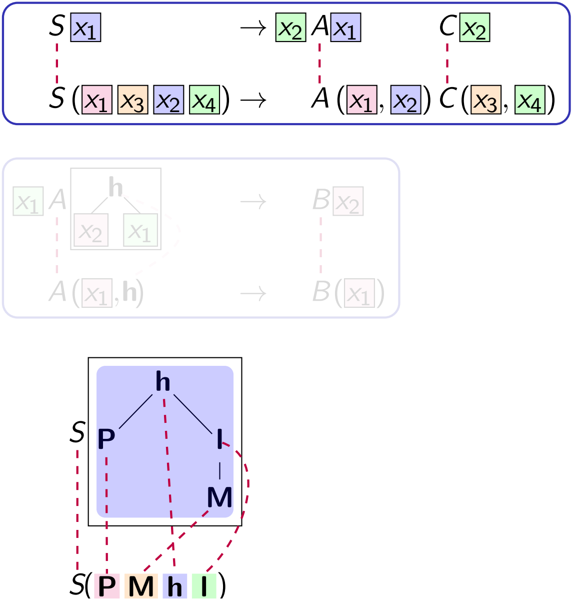
(LCFRS,sDCP)-hybrid grammar



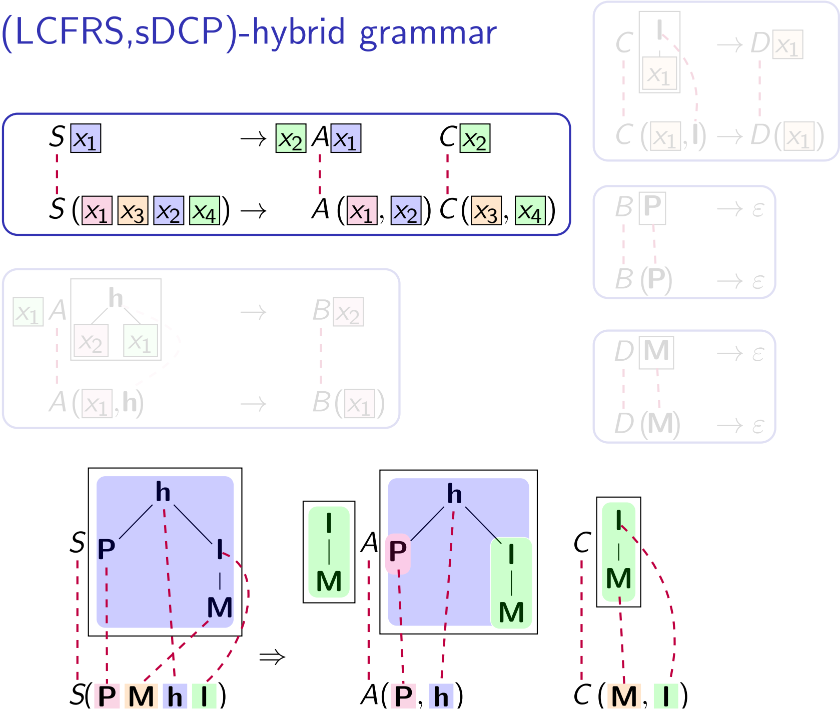
(LCFRS,sDCP)-hybrid grammar



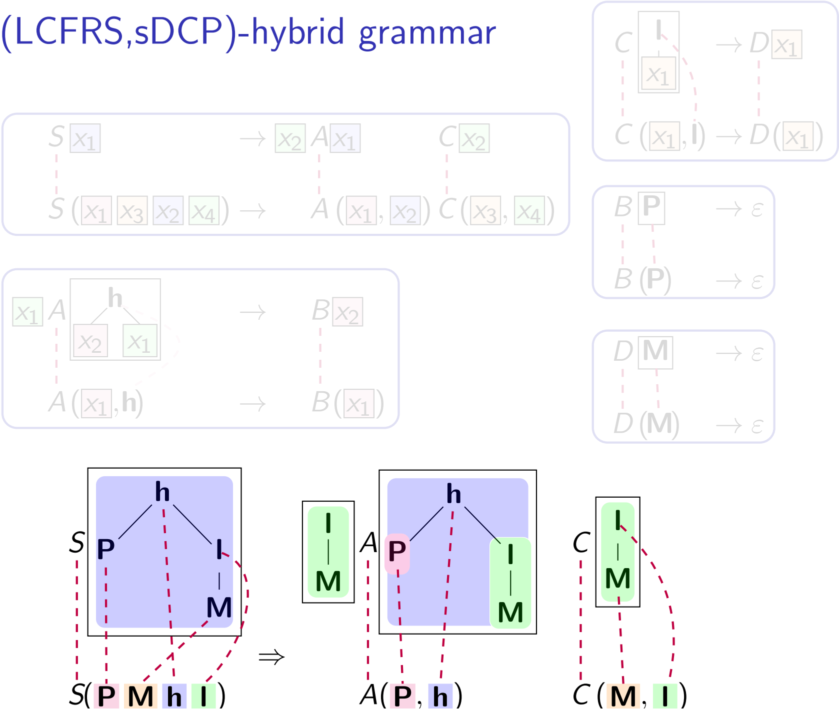
(LCFRS,sDCP)-hybrid grammar



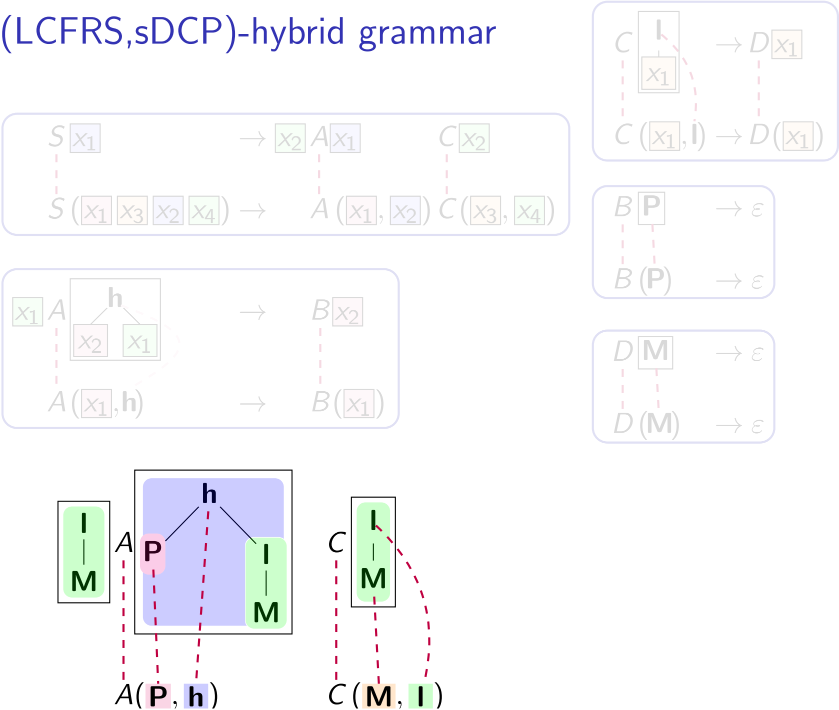
(LCFRS,sDCP)-hybrid grammar



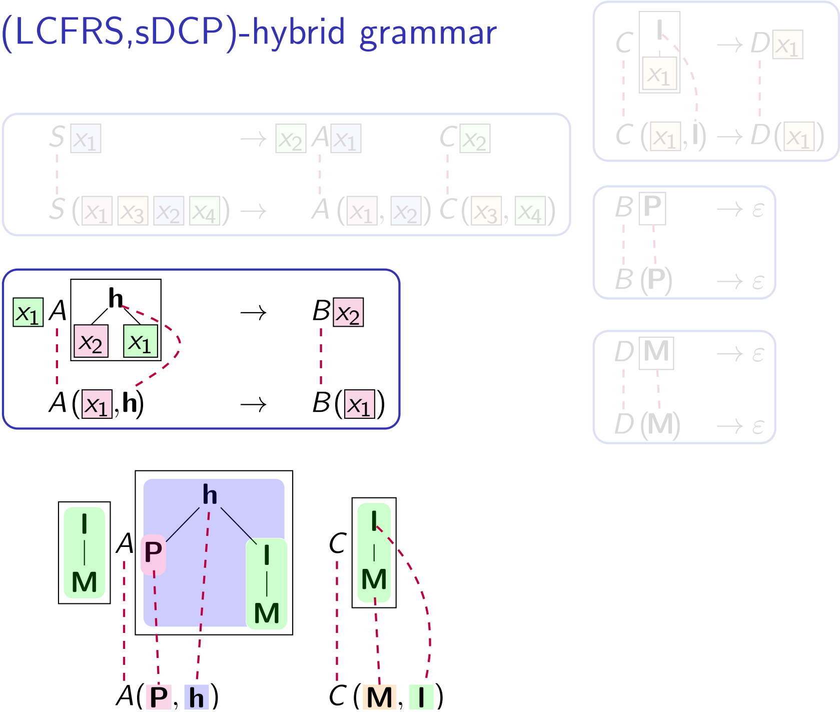
(LCFRS,sDCP)-hybrid grammar



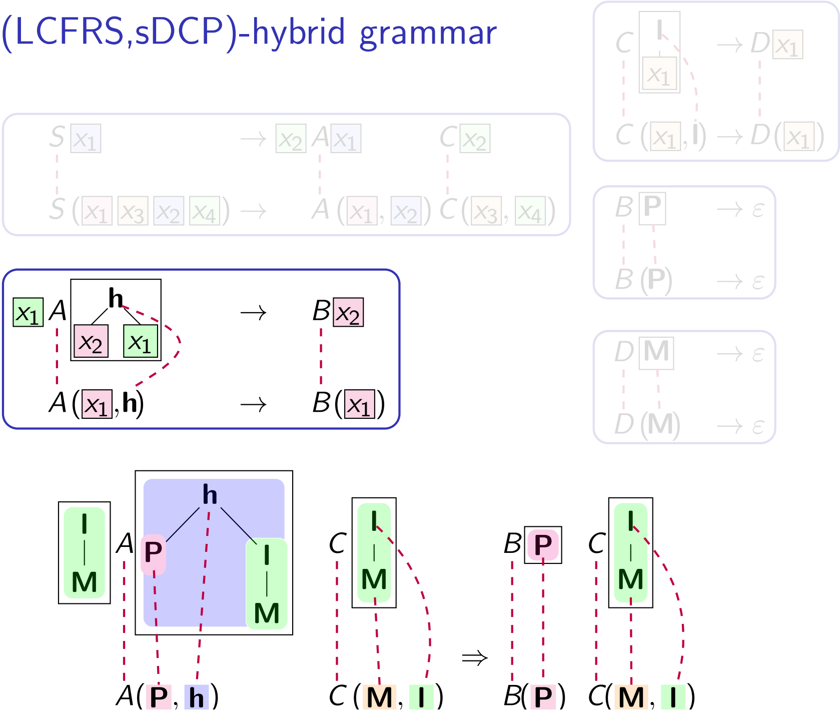
(LCFRS,sDCP)-hybrid grammar



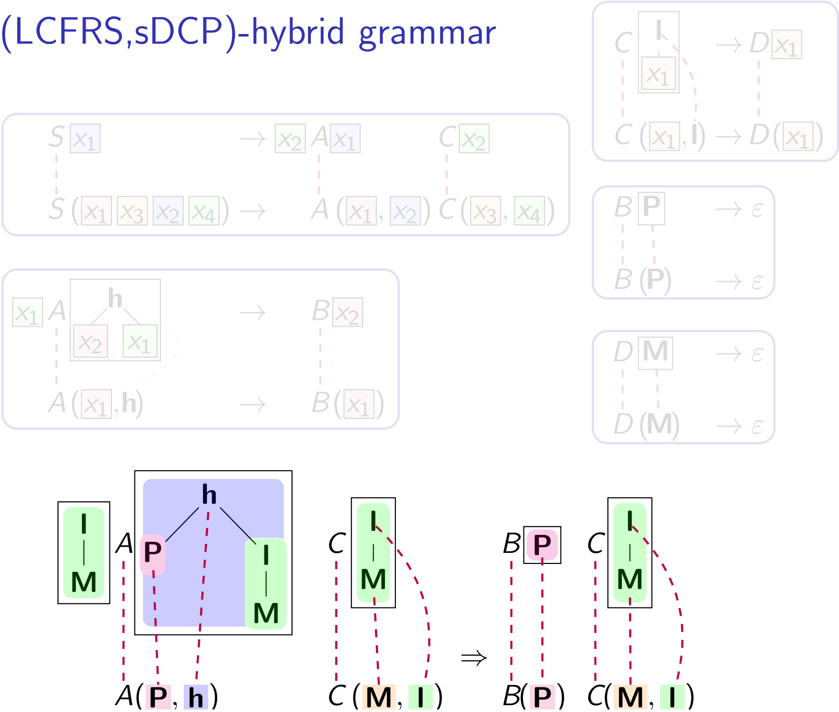
(LCFRS,sDCP)-hybrid grammar



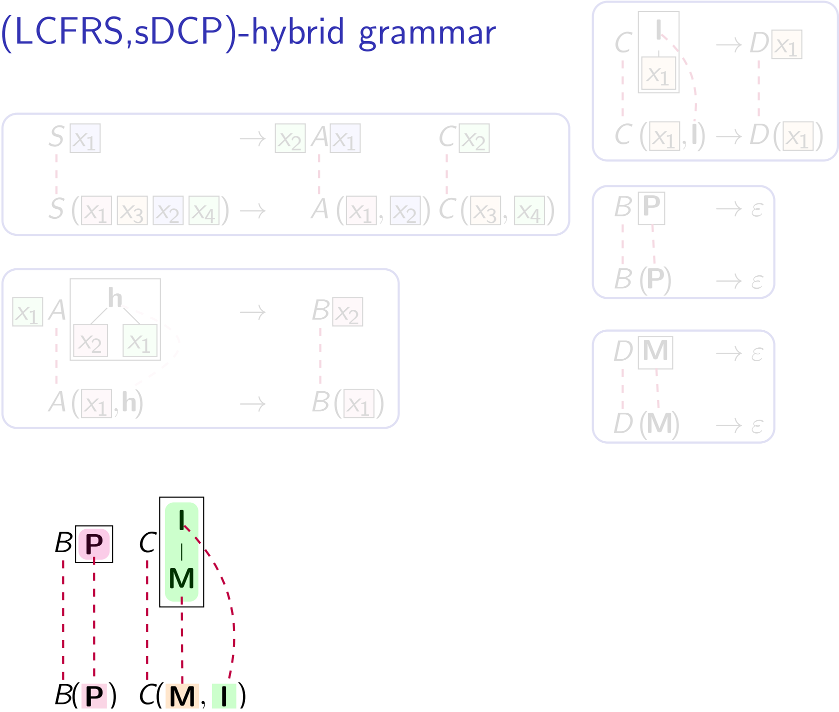
(LCFRS,sDCP)-hybrid grammar



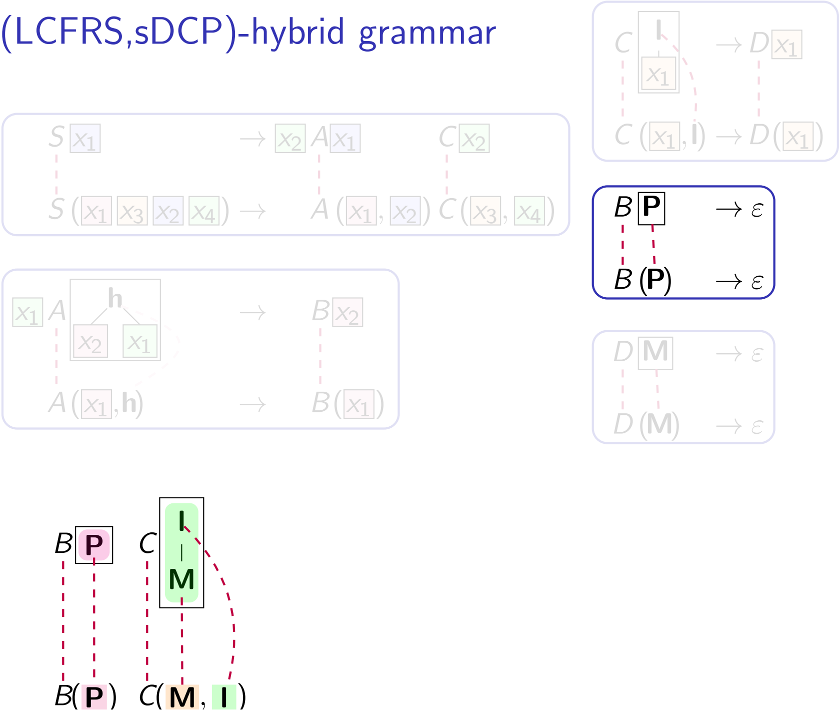
(LCFRS,sDCP)-hybrid grammar



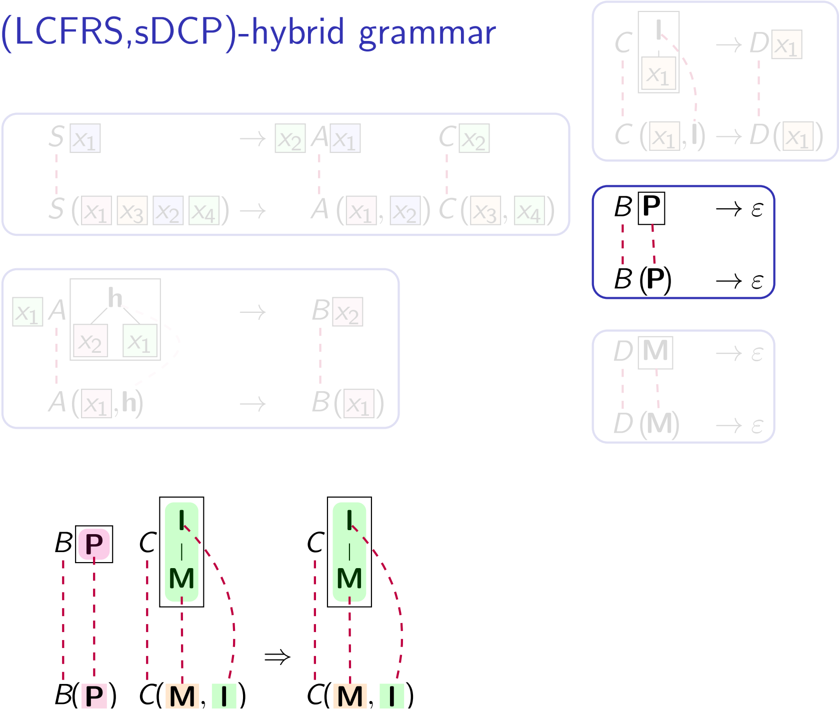
(LCFRS,sDCP)-hybrid grammar



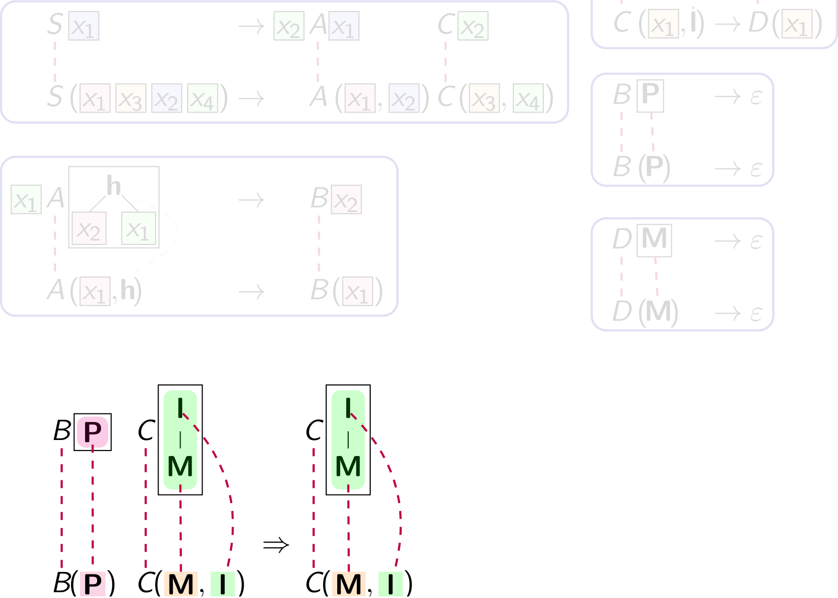
(LCFRS,sDCP)-hybrid grammar



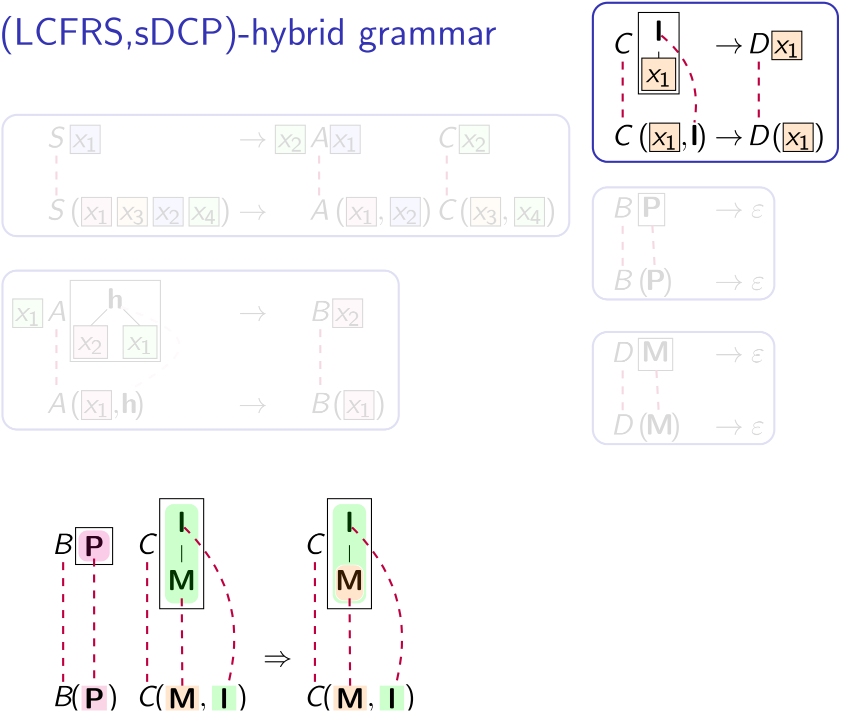
(LCFRS,sDCP)-hybrid grammar



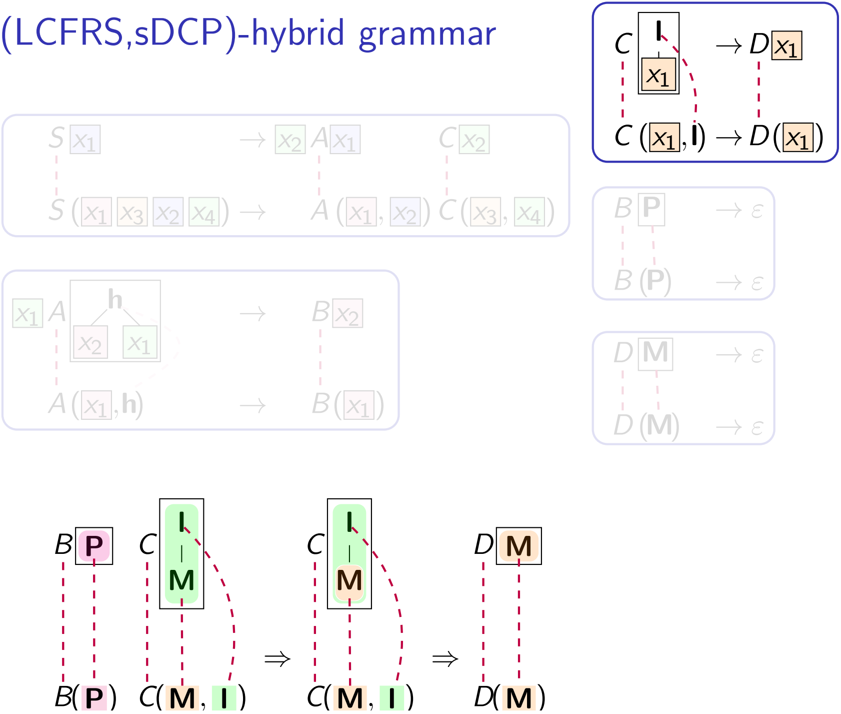
(LCFRS,sDCP)-hybrid grammar



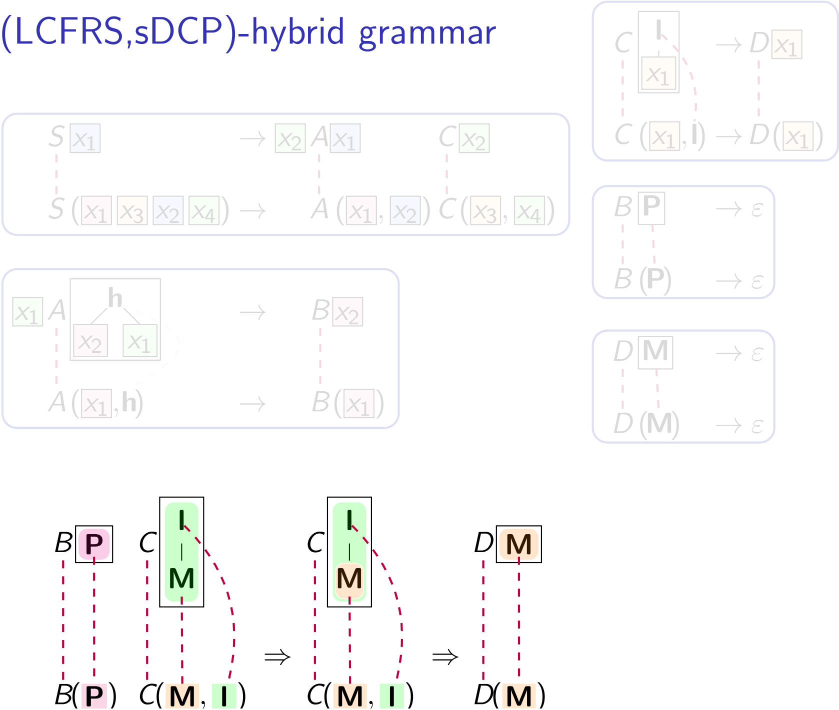
(LCFRS,sDCP)-hybrid grammar



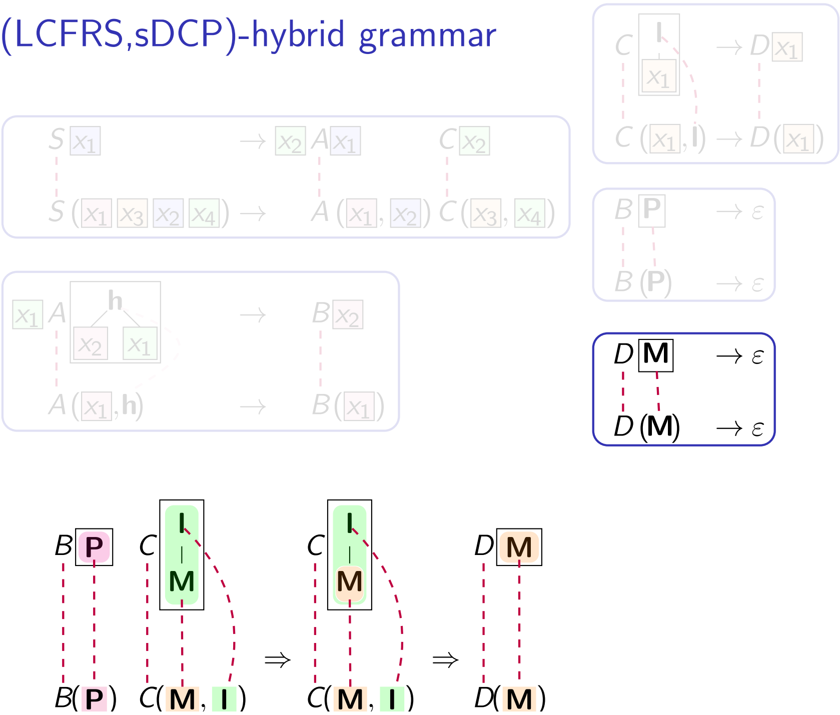
(LCFRS,sDCP)-hybrid grammar



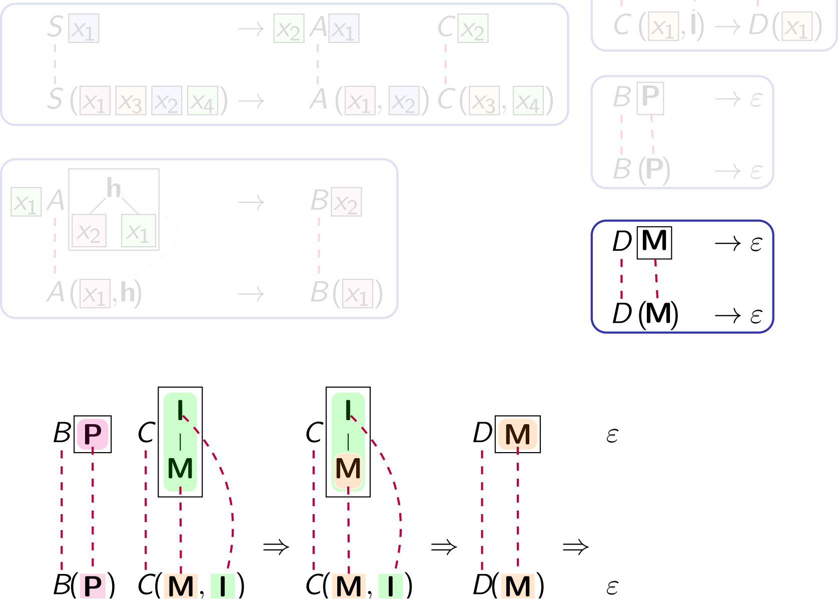
(LCFRS,sDCP)-hybrid grammar



(LCFRS,sDCP)-hybrid grammar



(LCFRS,sDCP)-hybrid grammar



1. hybrid trees and (LCFRS,sDCP)-hybrid grammars

2. grammar induction

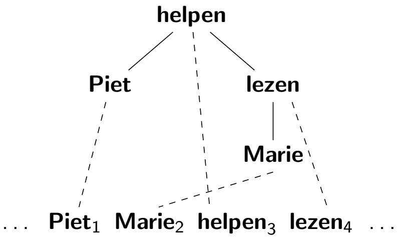
3. experiments

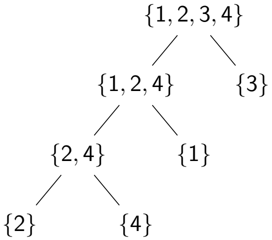
grammar induction

basic algorithm:

given: hybrid tree h and recursive partitioning π of $\mathrm{str}(h)$.

task: construct hybrid grammar which generates h according to π.



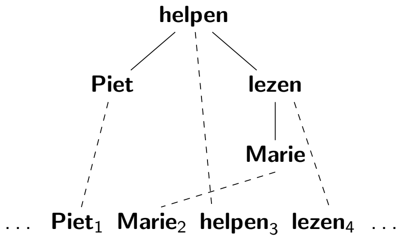


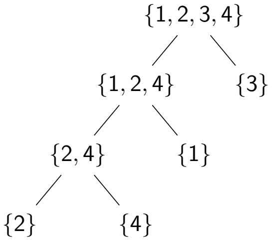
grammar induction

basic algorithm:

given: hybrid tree *h* and recursive partitioning π of $\mathrm{str}(h)$.

task: construct hybrid grammar which generates *h* according to π.

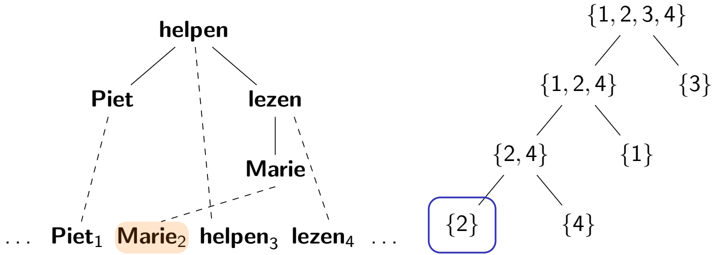




grammar induction

basic algorithm:

given: hybrid tree h and recursive partitioning π of $\mathrm{str}(h)$.
task: construct hybrid grammar which generates h according to π.



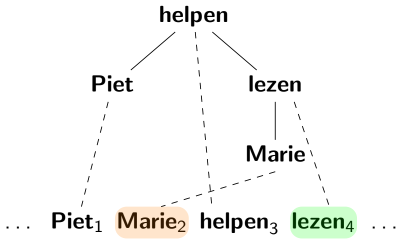
$\{2\}\,(\mathbf{M}) \rightarrow \varepsilon$

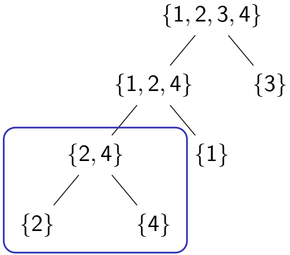
grammar induction

basic algorithm:

given: hybrid tree h and recursive partitioning π of $\mathrm{str}(h)$.

task: construct hybrid grammar which generates h according to π.





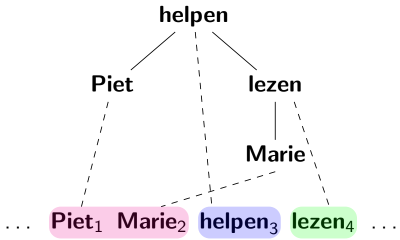
$\{2\}\,(\mathbf{M}) \rightarrow \varepsilon \qquad \{2,4\}(\boxed{x_1}, \boxed{x_2}) \rightarrow \{2\}(\boxed{x_1})\ \ \{4\}(\boxed{x_2})$

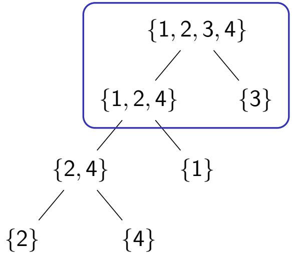
grammar induction

basic algorithm:

given: hybrid tree h and recursive partitioning π of $\mathrm{str}(h)$.

task: construct hybrid grammar which generates h according to π.





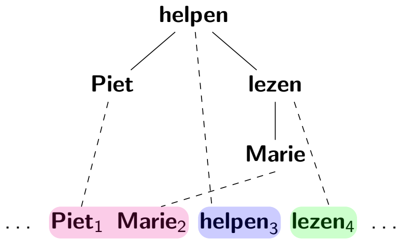
$\{2\}\,(\mathbf{M}) \to \varepsilon \qquad \{2,4\}(\boxed{x_1}, \boxed{x_2}) \to \{2\}(\boxed{x_1})\ \ \{4\}(\boxed{x_2})$

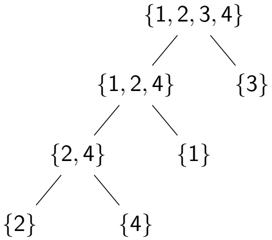
$\{1,2,3,4\}\,(\boxed{x_1}\,\boxed{x_3}\,\boxed{x_2}) \to \{1,2,4\}\,(\boxed{x_1}, \boxed{x_2})\ \ \{3\}\,(\boxed{x_3})$

grammar induction

basic algorithm:

given: hybrid tree h and recursive partitioning π of $\mathrm{str}(h)$.
task: construct hybrid grammar which generates h according to π.

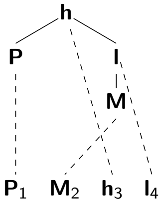


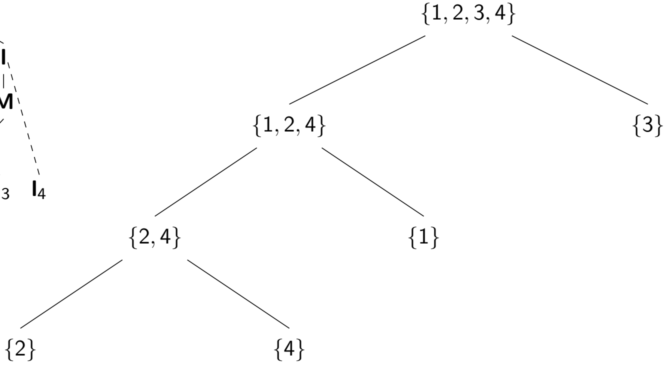


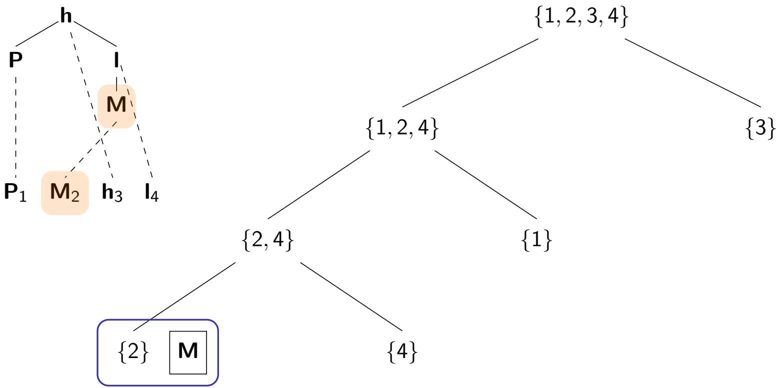
$\{2\}\,(\mathbf{M}) \to \varepsilon \qquad \{2,4\}(\boxed{x_1}, \boxed{x_2}) \to \{2\}(\boxed{x_1}) \;\; \{4\}(\boxed{x_2})$

$\{1,2,3,4\}\,(\boxed{x_1}\,\boxed{x_3}\,\boxed{x_2}) \to \{1,2,4\}\,(\boxed{x_1}, \boxed{x_2}) \;\; \{3\}\,(\boxed{x_3})$

sDCP: fold tree consistently onto recursive partitioning

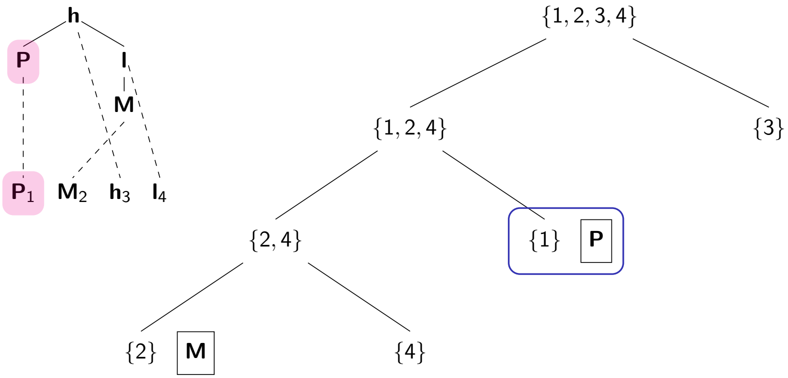

h
P
I
M
P_1
M_2
h_3
I_4


{1, 2, 3, 4}
{1, 2, 4}
{3}
{2, 4}
{1}
{2}
{4}



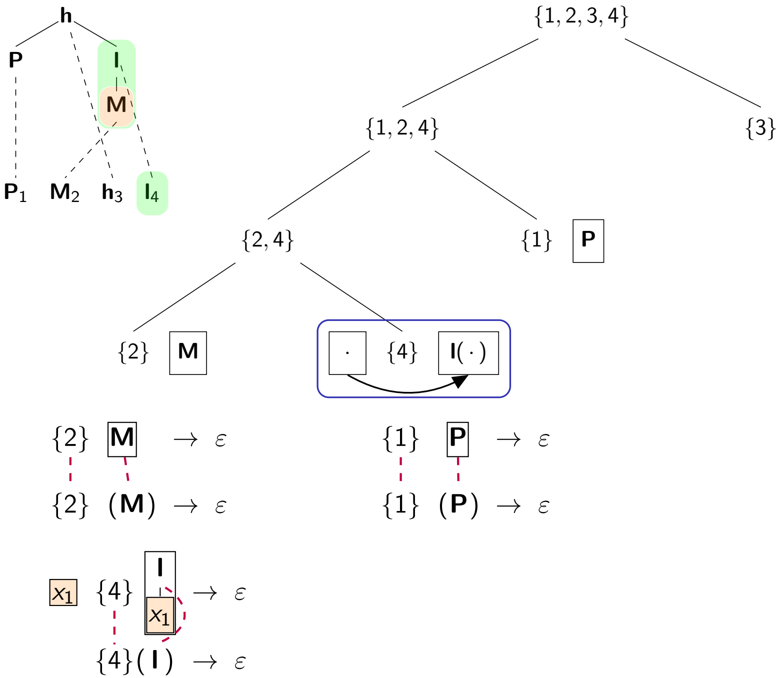
$$\{2\}\ \boxed{\mathbf{M}} \rightarrow \varepsilon$$

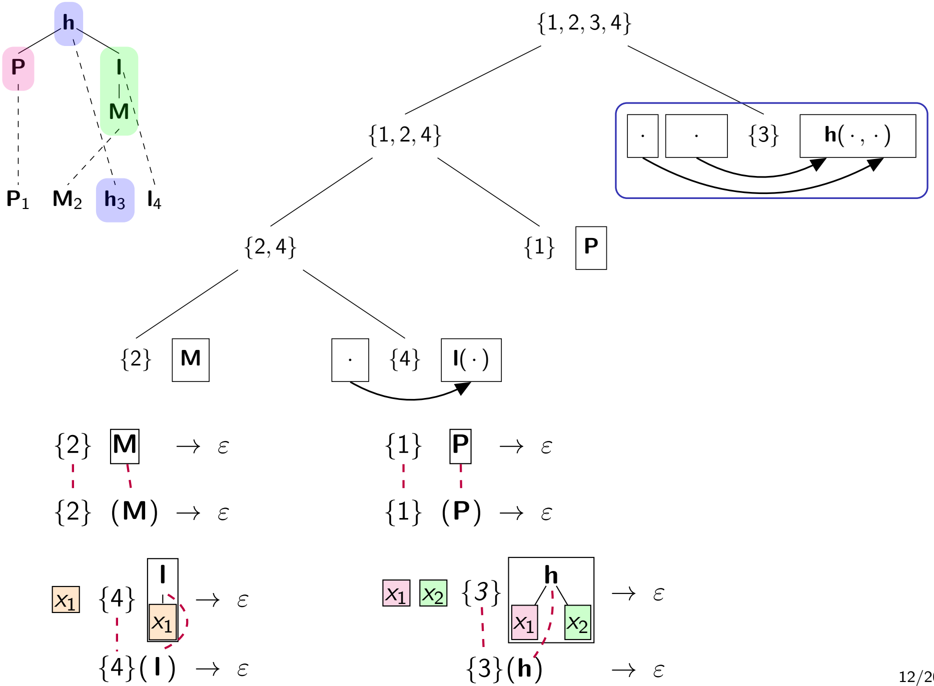
$$\{2\}\ (\mathbf{M}) \rightarrow \varepsilon$$



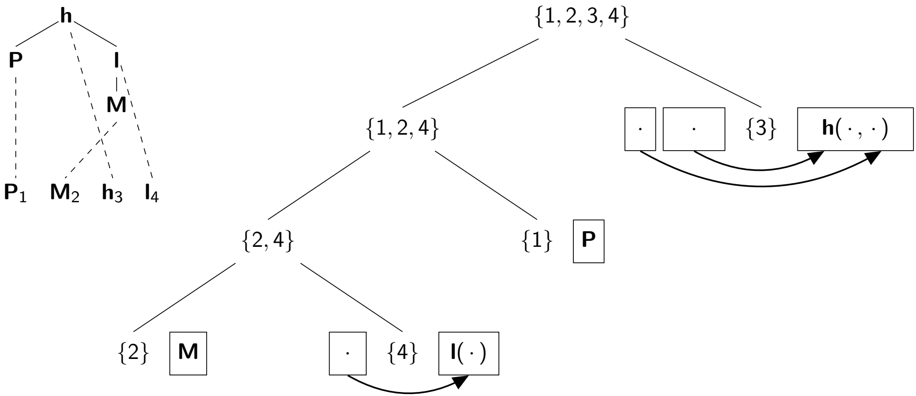
$$\{2\}\ \boxed{\mathbf{M}} \rightarrow \varepsilon \qquad \{1\}\ \boxed{\mathbf{P}} \rightarrow \varepsilon$$

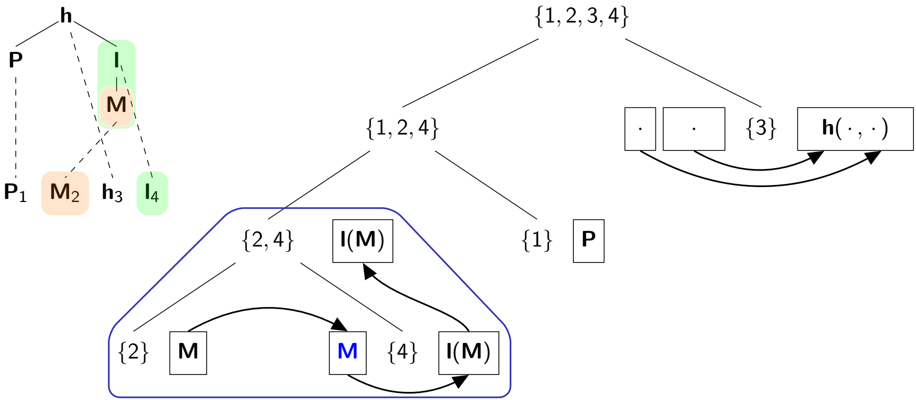
$$\{2\}\ (\mathbf{M}) \rightarrow \varepsilon \qquad \{1\}\ (\mathbf{P}) \rightarrow \varepsilon$$


h
P
I
M
P1
M2
h3
I4
{1, 2, 3, 4}
{1, 2, 4}
{3}
{2, 4}
{1}
P
{2}
M
·
{4}
I(·)
{2} M → ε
{2} (M) → ε
{1} P → ε
{1} (P) → ε
x1 {4} I x1 → ε
{4}(I) → ε



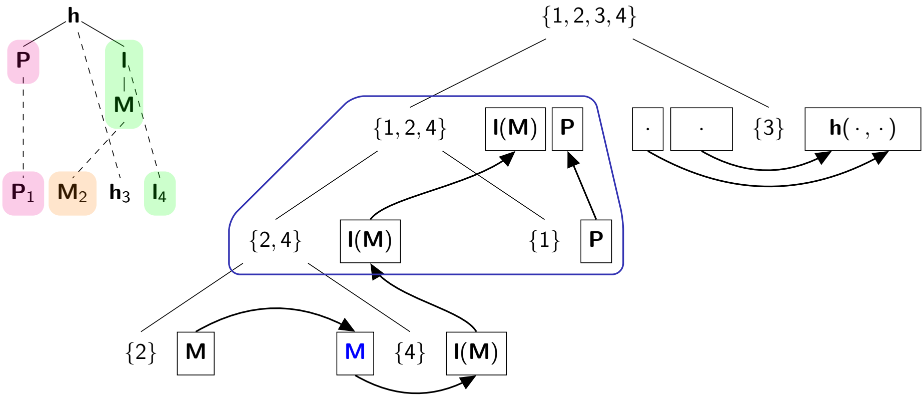
h
P
I
M
P1
M2
h3
I4
{1, 2, 3, 4}
{1, 2, 4}
{3}
h(· , ·)
{2, 4}
{1}
P
{2}
M
{4}
I(·)
{2} M → ε
{2} (M) → ε
{1} P → ε
{1} (P) → ε
x1 {4} I x1 → ε
{4}(I) → ε
x1 x2 {3} h x1 x2 → ε
{3}(h) → ε


h
P
I
M
P1 M2 h3 I4
{1, 2, 3, 4}
{1, 2, 4}
· · {3} h(· , ·)
{2, 4}
{1} P
{2} M
· {4} I(·)



$$\{2,4\}\ \boxed{x_2} \quad\rightarrow\quad \{2\}\boxed{x_1} \qquad \boxed{x_1}\ \{4\}\ \boxed{x_2}$$

$$\{2,4\}\ (\boxed{x_1},\boxed{x_2}) \quad\rightarrow\quad \{2\}(\boxed{x_1}) \qquad \{4\}\ (\boxed{x_2})$$

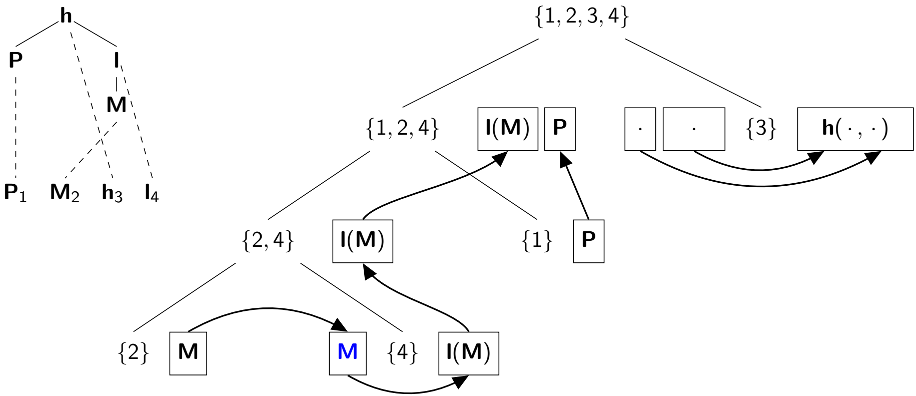


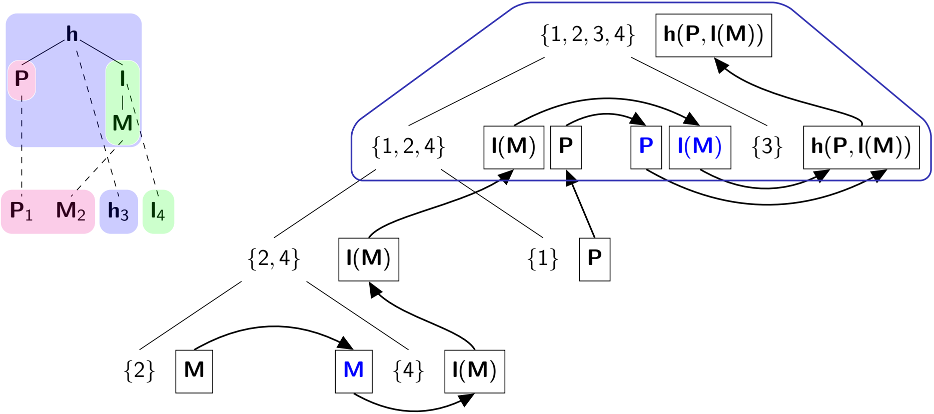
$\{2,4\}\ \boxed{x_2} \quad \rightarrow \quad \{2\}\boxed{x_1} \quad \boxed{x_1}\ \{4\}\ \boxed{x_2}$

$\{2,4\}\ (\boxed{x_1}, \boxed{x_2}) \quad \rightarrow \quad \{2\}(\boxed{x_1}) \quad \{4\}\ (\boxed{x_2})$

$\{1,2,4\}\ \boxed{x_1}\ \boxed{x_2} \quad \rightarrow \quad \{2,4\}\ \boxed{x_1} \quad \{1\}\ \boxed{x_2}$

$\{1,2,4\}\ (\boxed{x_3}\ \boxed{x_1}, \boxed{x_2}) \quad \rightarrow \quad \{2,4\}\ (\boxed{x_1}, \boxed{x_2})\ \{1\}\ (\boxed{x_3})$


h
P
I
M
P1
M2
h3
I4
{1, 2, 3, 4}
{1, 2, 4}
I(M)
P
·
·
{3}
h(· , ·)
{2, 4}
I(M)
{1}
P
{2}
M
M
{4}
I(M)



$$\{1,2,3,4\}\ \boxed{x_3} \quad \rightarrow \quad \{1,2,4\}\ \boxed{x_1}\ \boxed{x_2} \qquad \boxed{x_2}\ \boxed{x_1}\ \{3\}\ \boxed{x_3}$$

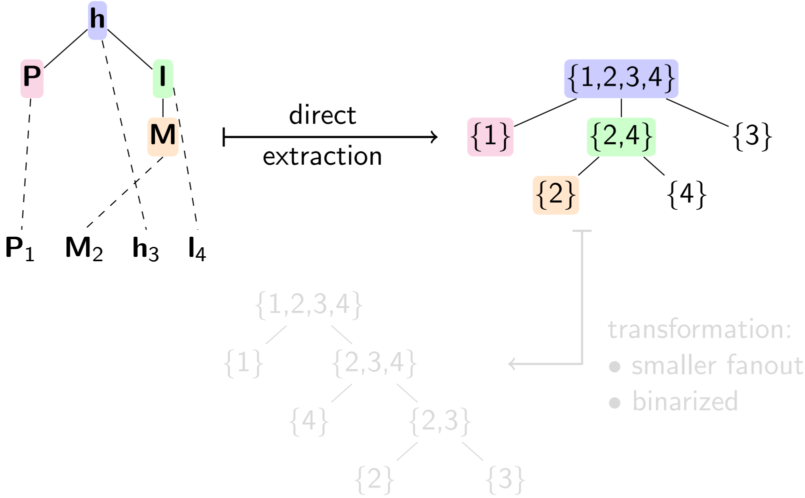
$$\{1,2,3,4\}\ (\boxed{x_1}\ \boxed{x_3}\ \boxed{x_2}) \quad \rightarrow \quad \{1,2,4\}\ (\boxed{x_1}, \boxed{x_2}) \qquad \{3\}\ (\boxed{x_3})$$

transforming recursive partitionings

the recursive partitioning π determines
the fanout and the rank of the induced grammar

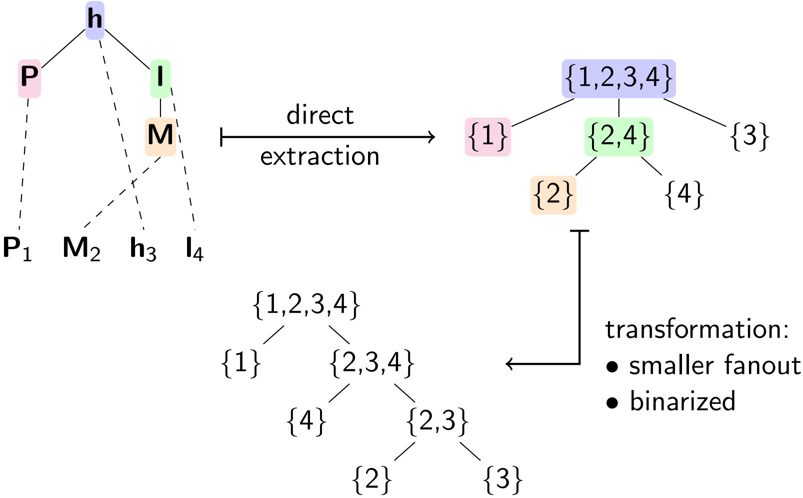
transforming recursive partitionings

the recursive partitioning π determines
the fanout and the rank of the induced grammar



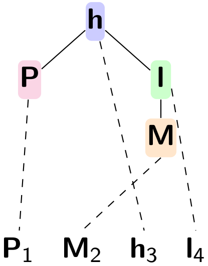
transforming recursive partitionings

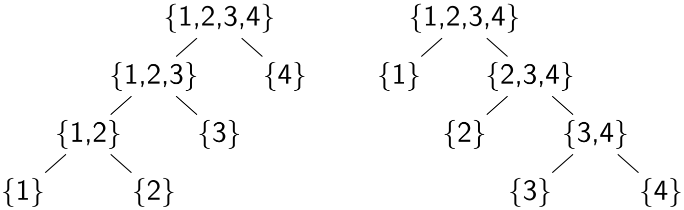
the recursive partitioning π determines
the fanout and the rank of the induced grammar



transforming recursive partitionings

the recursive partitioning π determines
the fanout and the rank of the induced grammar





grammar induction on a corpus

corpus of
hybrid trees

grammar induction on a corpus

corpus of
hybrid trees

1. choice of recursive partitioning π_i

grammar induction on a corpus

corpus of
hybrid trees

1. choice of recursive partitioning π_i
2. choice of nonterminals

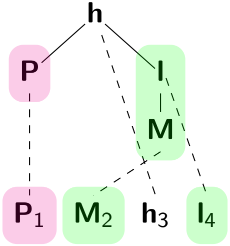
strict labeling

grammar induction on a corpus

corpus of
hybrid trees

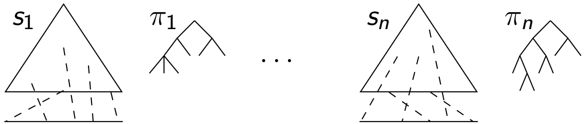
1. choice of recursive partitioning π_i
2. choice of nonterminals

	strict labeling
{1,2,4}	[**P**, **I**]



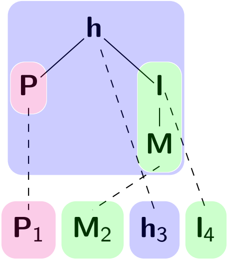
grammar induction on a corpus

corpus of
hybrid trees



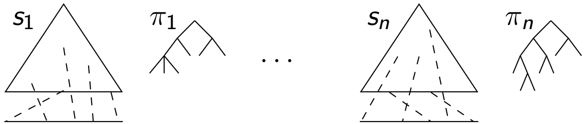
1. choice of recursive partitioning π_i
2. choice of nonterminals

	strict labeling
{1,2,4}	**[P, I]**
{3}	**[P, I | h]**



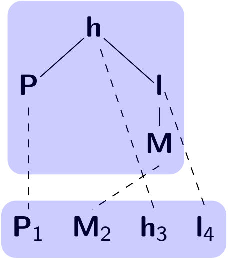
grammar induction on a corpus

corpus of hybrid trees



1. choice of recursive partitioning π_i
2. choice of nonterminals

	strict labeling
{1,2,4}	$[\mathbf{P}, \mathbf{I}]$
{3}	$[\mathbf{P}, \mathbf{I} \mid \mathbf{h}]$
{1,2,3,4}	$[\mathbf{h}]$

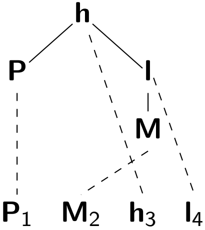


grammar induction on a corpus

corpus of
hybrid trees

1. choice of recursive partitioning π_i
2. choice of nonterminals

	strict labeling	child labeling
{1,2,4}	[**P**, **l**]	[children-of(**h**)]
{3}	[**P**, **l** | **h**]	[children-of(**h**) | **h**]
{1,2,3,4}	[**h**]	[**h**]

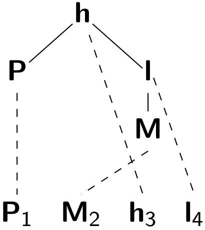


grammar induction on a corpus

corpus of
hybrid trees

1. choice of recursive partitioning π_i
2. choice of nonterminals

	strict labeling	child labeling
{1,2,4}	[**P**, **I**]	[children-of(**h**)]
{3}	[**P**, **I** | **h**]	[children-of(**h**) | **h**]
{1,2,3,4}	[**h**]	[**h**]



3. weighting productions by relative frequency estimation

1. hybrid trees and (LCFRS,sDCP)-hybrid grammars

2. grammar induction

3. experiments

experiment setup

prototypical implementation in python

corpora

- TIGER (German)
- NEGRA (German)
- METU-Sabanci Turkish Treebank
- Slovene Dependency Treebank (SDT)

split into
training set + test set

modifications

experiment setup

prototypical implementation in python

corpora

- TIGER (German)
- NEGRA (German)
- METU-Sabanci Turkish Treebank
- Slovene Dependency Treebank (SDT)

split into
training set + test set

modifications

experiment setup

prototypical implementation in python

corpora

- TIGER (German)
- NEGRA (German)
- METU-Sabanci Turkish Treebank
- Slovene Dependency Treebank (SDT)

split into
training set + test set

modifications

experiment setup

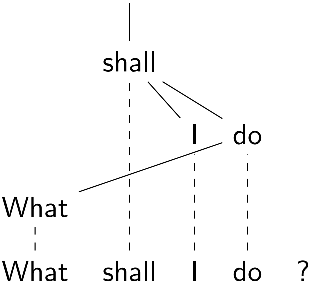
prototypical implementation in python

corpora

- TIGER (German)
- NEGRA (German)
- METU-Sabanci Turkish Treebank
- Slovene Dependency Treebank (SDT)

split into
training set + test set

modifications



experiment setup

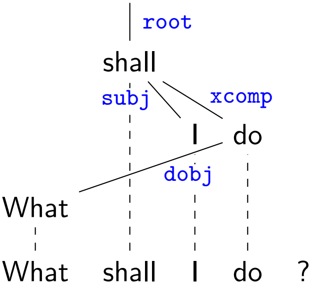
prototypical implementation in python

corpora

- TIGER (German)
- NEGRA (German)
- METU-Sabanci Turkish Treebank
- Slovene Dependency Treebank (SDT)

split into
training set + test set

modifications



experiment setup

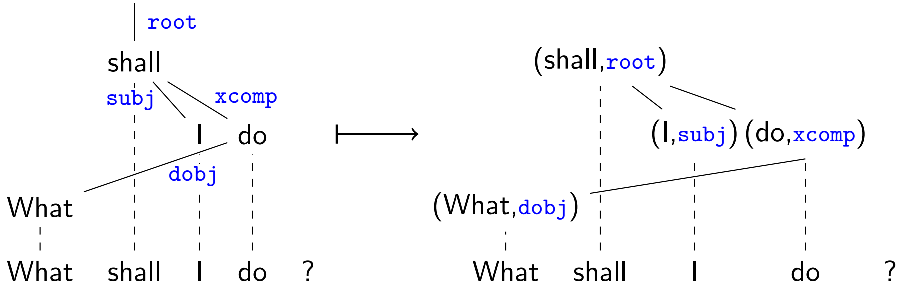
prototypical implementation in python

corpora

- TIGER (German)
- NEGRA (German)
- METU-Sabanci Turkish Treebank
- Slovene Dependency Treebank (SDT)

split into
training set + test set

modifications



experiment setup

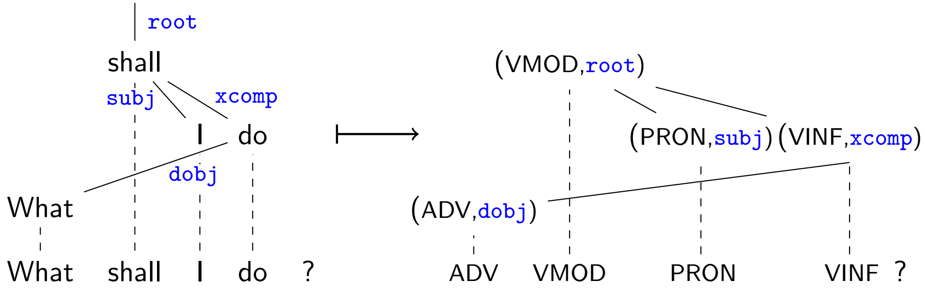
prototypical implementation in python

corpora

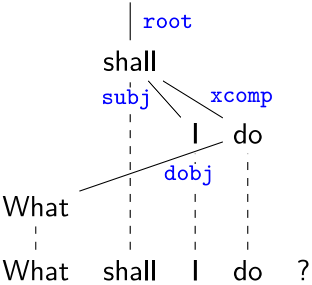
- TIGER (German)
- NEGRA (German)
- METU-Sabanci Turkish Treebank
- Slovene Dependency Treebank (SDT)

split into
training set + test set

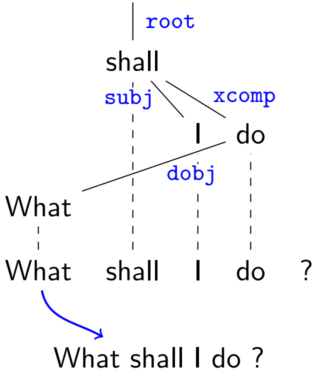
modifications



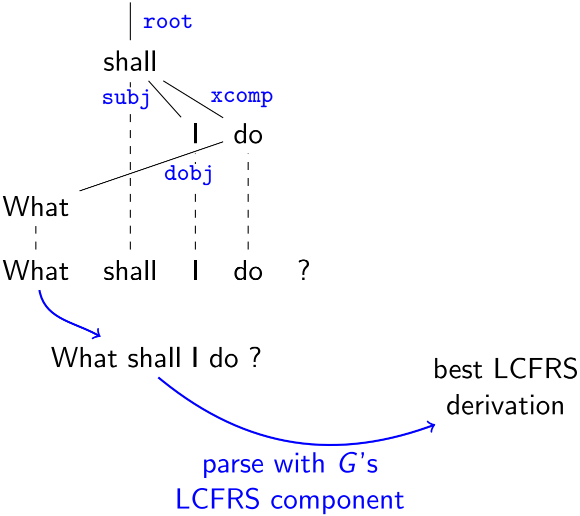
dependency parsing and evaluation



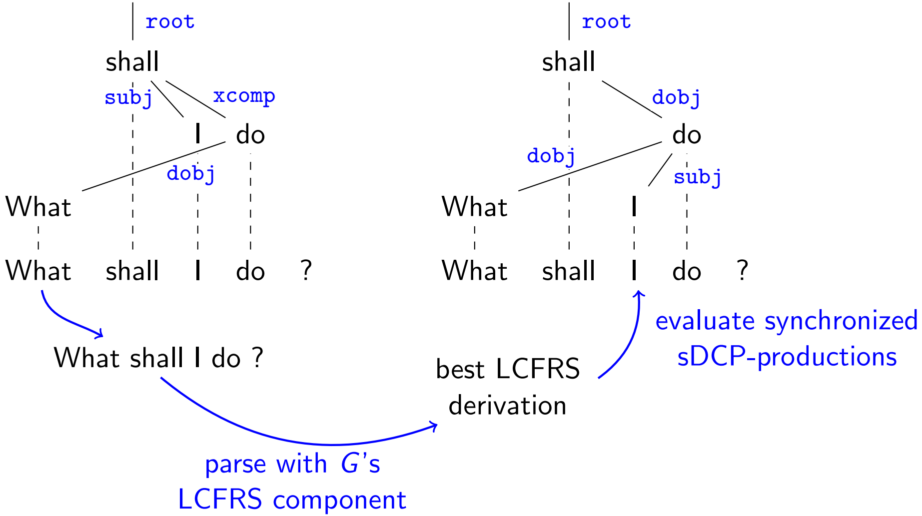
dependency parsing and evaluation



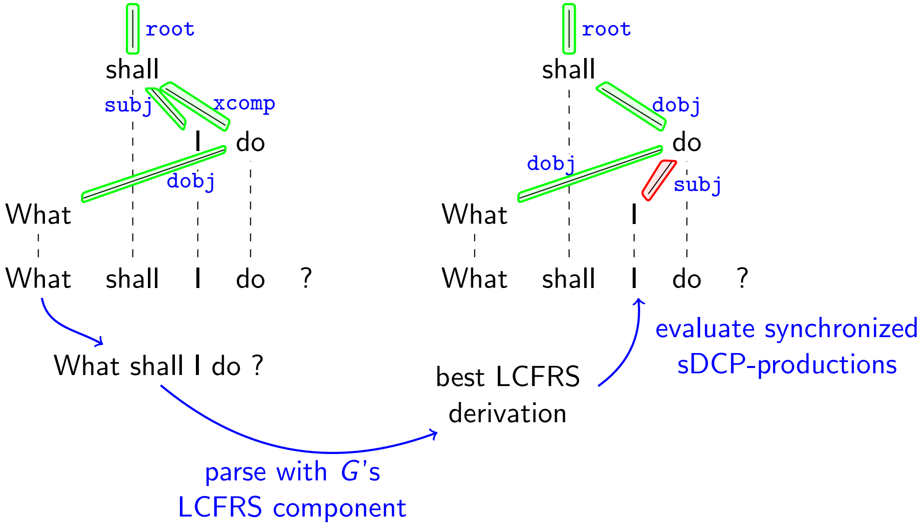
dependency parsing and evaluation



dependency parsing and evaluation

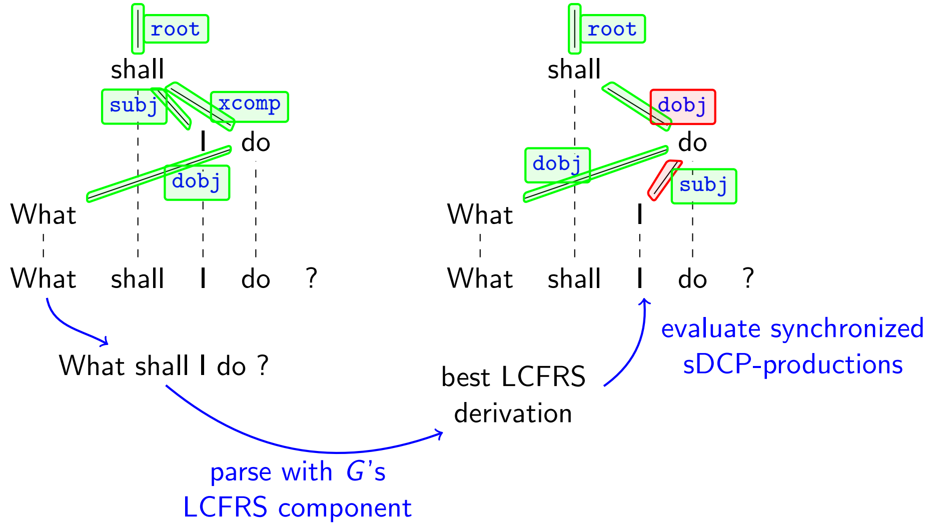


dependency parsing and evaluation



dependency parsing and evaluation

Labeled Attachment Score (LAS): 2/4



Results

corpus
TIGER
NEGRA*
METU
SDT

* with punctuation

Results

corpus	LAS reference (lex.)
TIGER	87.3
NEGRA*	82.0
METU	65.7
SDT	73.4

* with punctuation

Results

corpus	LAS reference (lex.)	LAS CF baseline
TIGER	87.3	80.4
NEGRA*	82.0	74.5
METU	65.7	39.9
SDT	73.4	53.7

* with punctuation

Results

corpus	LAS reference (lex.)	LAS CF baseline
TIGER	87.3	80.4
NEGRA*	82.0	74.5
METU	65.7	39.9
SDT	73.4	53.7

* with punctuation

lexicalization often yields an improvement of 5-10 % in the LAS

Results

corpus	LAS reference (lex.)	LAS CF baseline	gain fanout 2
TIGER	87.3	80.4	0.5–2.0
NEGRA*	82.0	74.5	?
METU	65.7	39.9	1.0-2.0
SDT	73.4	53.7	?

* with punctuation

lexicalization often yields an improvement of 5-10 % in the LAS

Results

corpus	LAS reference (lex.)	LAS CF baseline	gain fanout 2	loss l/r-branching
TIGER	87.3	80.4	0.5–2.0	1.0–2.0
NEGRA*	82.0	74.5	?	2.0–3.0
METU	65.7	39.9	1.0-2.0	2.0–3.0
SDT	73.4	53.7	?	1.0–2.0

* with punctuation

lexicalization often yields an improvement of 5-10 % in the LAS

Results

corpus	LAS reference (lex.)	LAS CF baseline	gain fanout 2	loss l/r-branching
TIGER	87.3	80.4	0.5–2.0	1.0–2.0
NEGRA*	82.0	74.5	?	2.0–3.0
METU	65.7	39.9	1.0-2.0	2.0–3.0
SDT	73.4	53.7	?	1.0–2.0

* with punctuation

lexicalization often yields an improvement of 5-10 % in the LAS

trade-off **and** separation of parsing complexity and quality

conclusion

- (LCFRS,sDCP)-hybrid grammars can be used for parsing of
  (discontinuous phrase structures and)
  non-projective dependency structures
- high-level parameterizable framework for grammar induction
  (recursive partitionings, labeling strategies)
- experimental evaluation

outlook

- lexicalization, Markovization, latent variables, etc.
- performance of an optimized (FSA*, sDCP)-hybrid grammar?
  * LR-CFG, LR-LCFRS, ...
- hybrid grammars as generative model
- other combinations of grammars
  bitransformation characterization

conclusion

- (LCFRS,sDCP)-hybrid grammars can be used for parsing of
  (discontinuous phrase structures and)
  non-projective dependency structures
- high-level parameterizable framework for grammar induction
  (recursive partitionings, labeling strategies)
- experimental evaluation

outlook

- lexicalization, Markovization, latent variables, etc.
- performance of an optimized (FSA*, sDCP)-hybrid grammar?
  * LR-CFG, LR-LCFRS, ...
- hybrid grammars as generative model
- other combinations of grammars
  bitransformation characterization

conclusion

- (LCFRS,sDCP)-hybrid grammars can be used for parsing of
  (discontinuous phrase structures and)
  non-projective dependency structures
- high-level parameterizable framework for grammar induction
  (recursive partitionings, labeling strategies)
- experimental evaluation

outlook

- lexicalization, Markovization, latent variables, etc.
- performance of an optimized (FSA*, sDCP)-hybrid grammar?
  * LR-CFG, LR-LCFRS, ...
- hybrid grammars as generative model
- other combinations of grammars
  bitransformation characterization

conclusion

- (LCFRS,sDCP)-hybrid grammars can be used for parsing of
  (discontinuous phrase structures and)
  non-projective dependency structures
- high-level parameterizable framework for grammar induction
  (recursive partitionings, labeling strategies)
- experimental evaluation

outlook

- lexicalization, Markovization, latent variables, etc.
- performance of an optimized (FSA*, sDCP)-hybrid grammar?
  * LR-CFG, LR-LCFRS, ...
- hybrid grammars as generative model
- other combinations of grammars
  bitransformation characterization

conclusion

- (LCFRS,sDCP)-hybrid grammars can be used for parsing of
  (discontinuous phrase structures and)
  non-projective dependency structures
- high-level parameterizable framework for grammar induction
  (recursive partitionings, labeling strategies)
- experimental evaluation

outlook

- lexicalization, Markovization, latent variables, etc.
- performance of an optimized (FSA*, sDCP)-hybrid grammar?
  * LR-CFG, LR-LCFRS, ...
- hybrid grammars as generative model
- other combinations of grammars
  bitransformation characterization

conclusion

- (LCFRS,sDCP)-hybrid grammars can be used for parsing of
  (discontinuous phrase structures and)
  non-projective dependency structures
- high-level parameterizable framework for grammar induction
  (recursive partitionings, labeling strategies)
- experimental evaluation

outlook

- lexicalization, Markovization, latent variables, etc.
- performance of an optimized (FSA*, sDCP)-hybrid grammar?
  * LR-CFG, LR-LCFRS, ...
- hybrid grammars as generative model
- other combinations of grammars
  bitransformation characterization

conclusion

- (LCFRS,sDCP)-hybrid grammars can be used for parsing of
  (discontinuous phrase structures and)
  non-projective dependency structures
- high-level parameterizable framework for grammar induction
  (recursive partitionings, labeling strategies)
- experimental evaluation

outlook

- lexicalization, Markovization, latent variables, etc.
- performance of an optimized (FSA*, sDCP)-hybrid grammar?
  * LR-CFG, LR-LCFRS, ...
- hybrid grammars as generative model
- other combinations of grammars
  bitransformation characterization

References

M.-J. Nederhof and H. Vogler. "Hybrid Grammars for Discontinuous Parsing". COLING. 2014, pp. 1370–1381.