Generic binarization for parsing and translation

Matthias Büchse
TU Dresden

Alexander Koller
Uni Potsdam

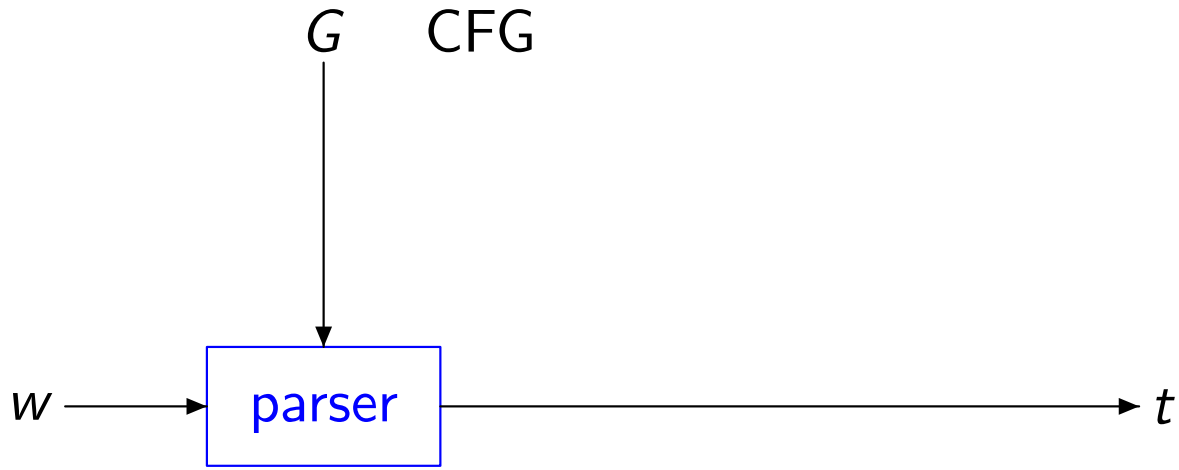
Heiko Vogler
TU Dresden

2013-08-05

Outline

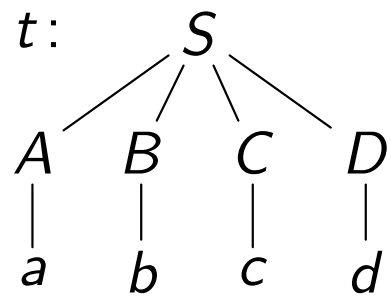
Outline

CFG binarization for parsing

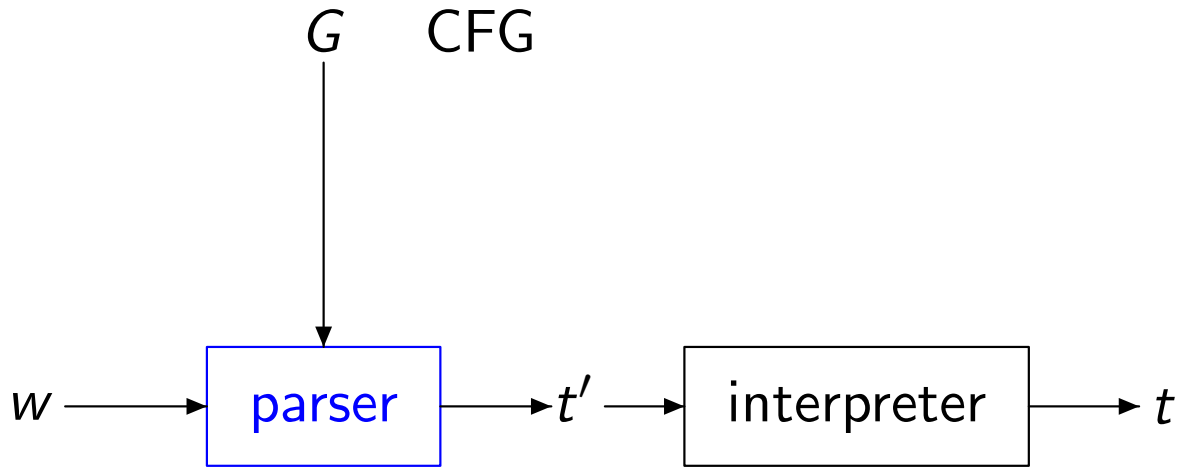


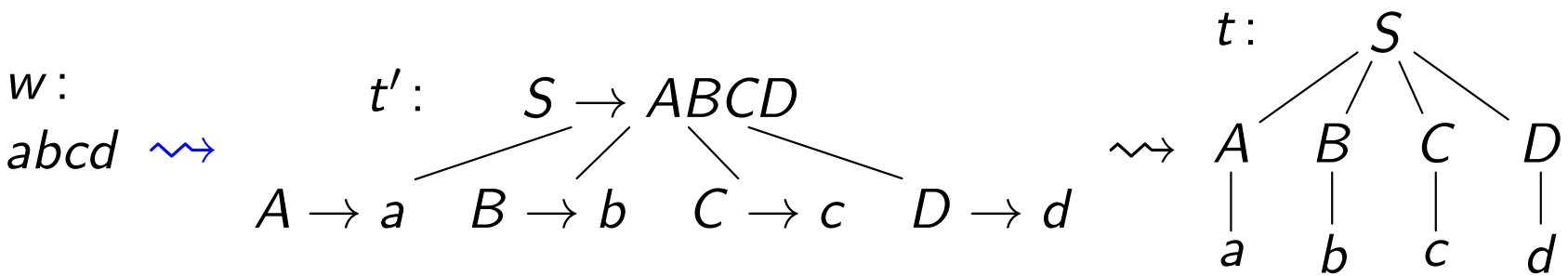
w :
$abcd$

$\rightsquigarrow$

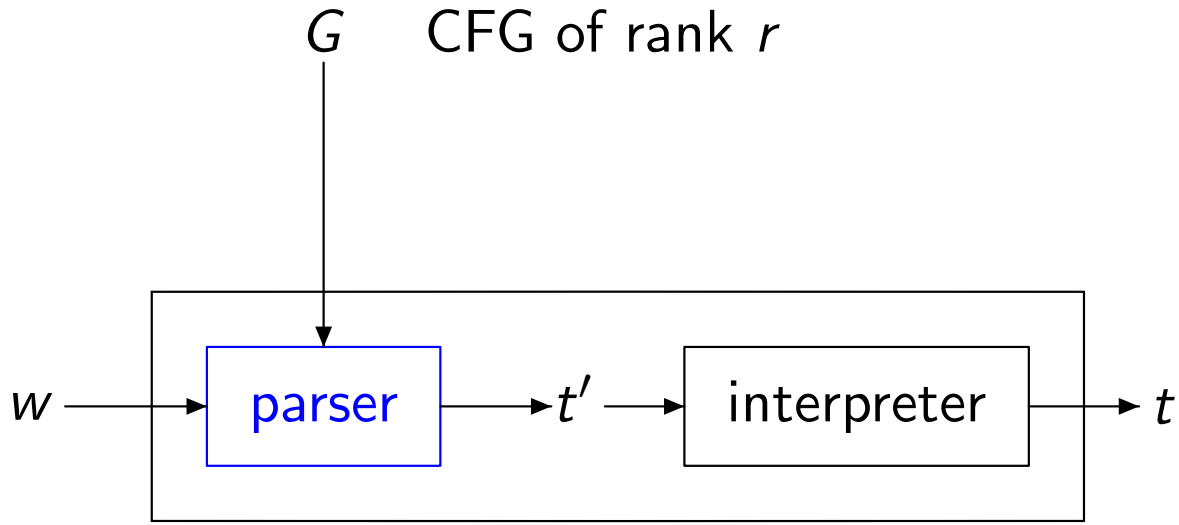


CFG binarization for parsing

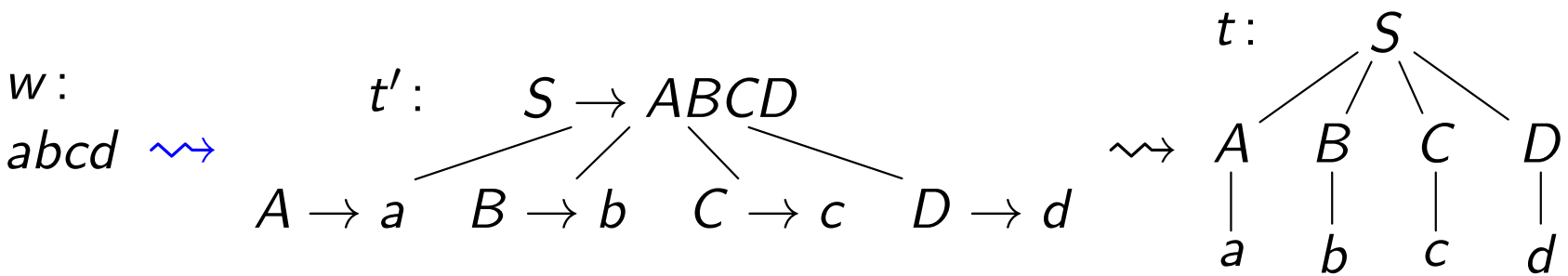




CFG binarization for parsing



$$O(|R| \cdot |w|^{r+1})$$



CFG binarization for parsing

CFG G: (rank 4)

$$S \to ABCD$$
$$A \to a$$
$$B \to b$$
$$C \to c$$
$$D \to d$$

CFG binarization for parsing

CFG G: (rank 4)

$S \to ABCD$

$A \to a$

$B \to b$

$C \to c$

$D \to d$

CFG binarization for parsing

CFG G: (rank 4)

$$S \to ABCD$$
$$A \to a$$
$$B \to b$$
$$C \to c$$
$$D \to d$$

$\rightsquigarrow$

CFG G': (rank 2)

$$S \to A[BCD]$$
$$[BCD] \to B[CD]$$
$$[CD] \to CD$$
$$\vdots$$

CFG binarization for parsing

CFG G: (rank 4)

$$S \to ABCD$$
$$A \to a$$
$$B \to b$$
$$C \to c$$
$$D \to d$$

$\rightsquigarrow$

CFG G': (rank 2)

$$S \to A[BCD]$$
$$[BCD] \to B[CD]$$
$$[CD] \to CD$$
$$\vdots$$

rule-by-rule binarization problem *given* G:

for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

CFG binarization for parsing

CFG G: (rank 4)

$$S \to ABCD$$
$$A \to a$$
$$B \to b$$
$$C \to c$$
$$D \to d$$

$\rightsquigarrow$

CFG G': (rank 2)

$$S \to A[BCD]$$
$$[BCD] \to B[CD]$$
$$[CD] \to CD$$
$$\vdots$$

rule-by-rule binarization problem *given* G:

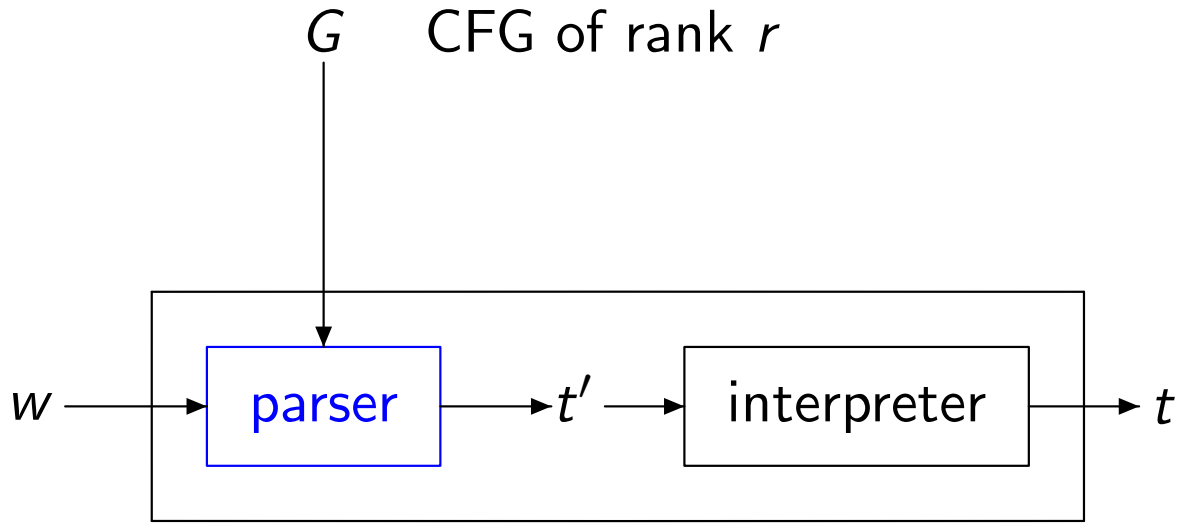
for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

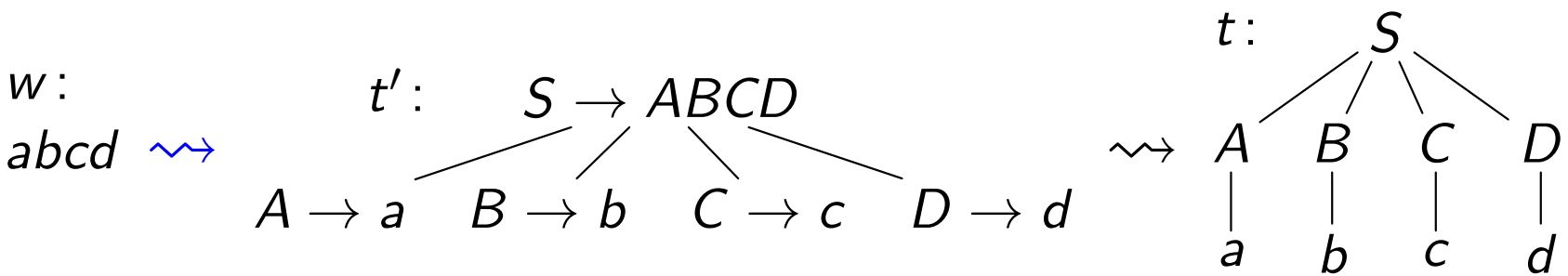
binarization problem:

convert given CFG G into equivalent CFG G' of rank 2

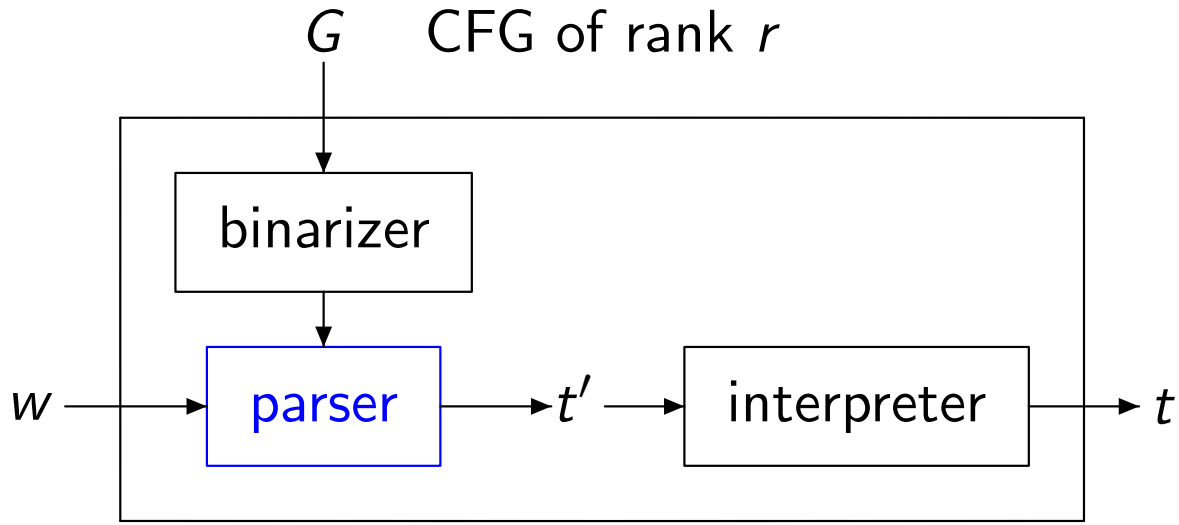
CFG binarization for parsing



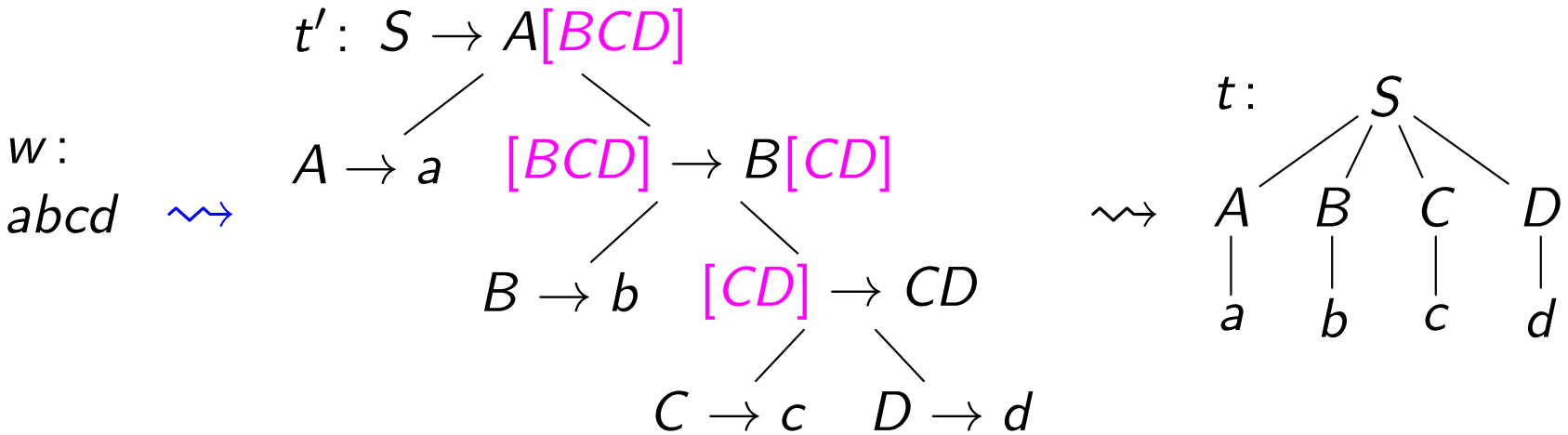
$$O(|R| \cdot |w|^{r+1})$$



CFG binarization for parsing

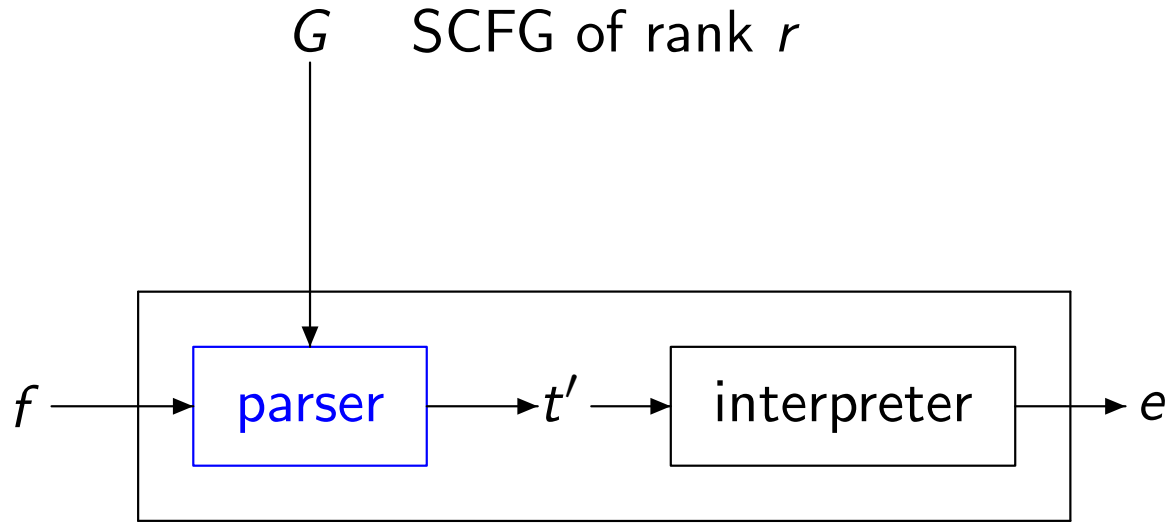


$$O(r \cdot |R| \cdot |w|^3)$$



Outline

SCFG binarization for translation



$$O(|R| \cdot |w|^{r+1})$$

f : $abcd$ $\rightsquigarrow$ t' : $S \to \langle ABCD, CADB \rangle$ ($A \to \langle a, a \rangle$, $B \to \langle b, b \rangle$, $C \to \langle c, c \rangle$, $D \to \langle d, d \rangle$) $\rightsquigarrow$ e : $cadb$

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$
$A \to \langle a, a \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

rule-by-rule binarization problem *given* G:

for each rule of rank > 2
find equivalent collection of rules of rank ≤ 2

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$
$A \to \langle a, a \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

rule-by-rule binarization problem *given* G:

for each rule of rank > 2
find equivalent collection of rules of rank ≤ 2

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$

$A \to \langle a, a \rangle$

$B \to \langle b, b \rangle$

$C \to \langle c, c \rangle$

$D \to \langle d, d \rangle$

$\rightsquigarrow$

SCFG G': (rank 2)

$S \to \langle A[BCD], CADB \rangle$ ↯

rule-by-rule binarization problem *given* G:

for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$
$A \to \langle a, a \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

$\rightsquigarrow$

SCFG G': (rank 2)

$S \to \langle A[BCD], CADB \rangle$ ↯
$S \to \langle [ABC]D, CADB \rangle$ ↯

rule-by-rule binarization problem *given* G:

for each rule of rank > 2
find equivalent collection of rules of rank ≤ 2

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$
$A \to \langle a, a \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

$\rightsquigarrow$

SCFG G': (rank 2)

$S \to \langle A[BCD], CADB \rangle$ ↯
$S \to \langle [ABC]D, CADB \rangle$ ↯
$S \to \langle [AB][CD], CADB \rangle$ ↯

rule-by-rule binarization problem *given* G:

for each rule of rank > 2
find equivalent collection of rules of rank ≤ 2

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$
$A \to \langle a, a \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

$\rightsquigarrow$

SCFG G': (rank 2)

$S \to \langle A[BCD], CADB \rangle$ ↯
$S \to \langle [ABC]D, CADB \rangle$ ↯
$S \to \langle [AB][CD], CADB \rangle$ ↯

rule-by-rule binarization problem *given* G:

for each rule of rank > 2
find equivalent collection of rules of rank ≤ 2

solution need not exist

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$
$A \to \langle a, a \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

$\rightsquigarrow$

SCFG G': (rank 2)

$S \to \langle A[BCD], CADB \rangle$ ↯
$S \to \langle [ABC]D, CADB \rangle$ ↯
$S \to \langle [AB][CD], CADB \rangle$ ↯

rule-by-rule binarization problem *given* G:

for each rule of rank > 2
find equivalent collection of rules of rank ≤ 2

solution need not exist
algorithm by Huang et al. (2009)

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$
$A \to \langle a, a \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

$\rightsquigarrow$

SCFG G': (rank 2)

$S \to \langle A[BCD], CADB \rangle$ ↯
$S \to \langle [ABC]D, CADB \rangle$ ↯
$S \to \langle [AB][CD], CADB \rangle$ ↯

rule-by-rule binarization problem *given* G:

for each rule of rank > 2
find equivalent collection of rules of rank ≤ 2

binarization problem:

convert given SCFG G into equivalent SCFG G' of rank 2

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$

$A \to \langle a, a \rangle$

$B \to \langle b, b \rangle$

$C \to \langle c, c \rangle$

$D \to \langle d, d \rangle$

$\rightsquigarrow$

SCFG G': (rank 0)

$S \to \langle abcd, cadb \rangle$

rule-by-rule binarization problem *given* G:

for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

binarization problem:

convert given SCFG G into equivalent SCFG G' of rank 2

SCFG binarization for translation

SCFG G: (rank 4)

$S \to \langle ABCD, CADB \rangle$

$A \to \langle a, a \rangle$

$B \to \langle b, b \rangle$

$C \to \langle c, c \rangle$

$D \to \langle d, d \rangle$

$\rightsquigarrow$

SCFG G': (rank 0)

$S \to \langle abcd, cadb \rangle$

rule-by-rule binarization problem *given* G:

for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

binarization problem:

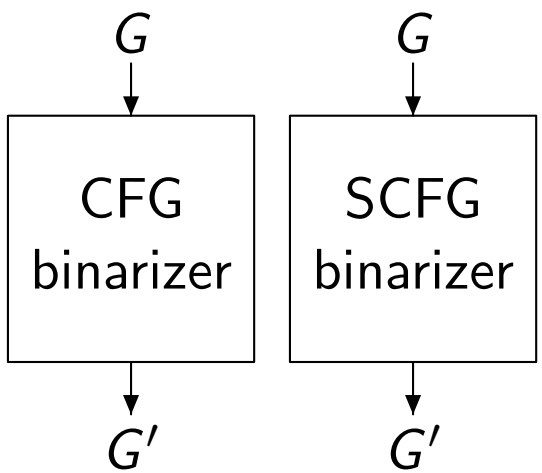
convert given SCFG G into equivalent SCFG G' of rank 2

again: solution need not exist (Aho and Ullman, 1969)

Outline

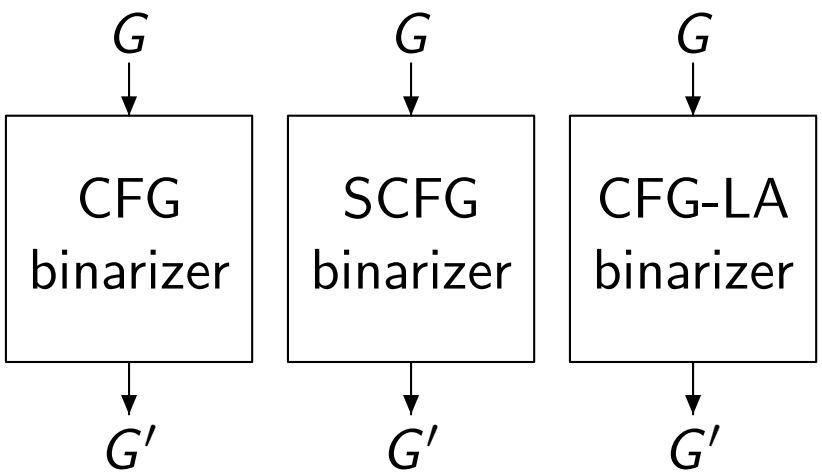
Further formalisms

- CFG (Chomsky)
- SCFG (Lewis and Stearns, 1966)



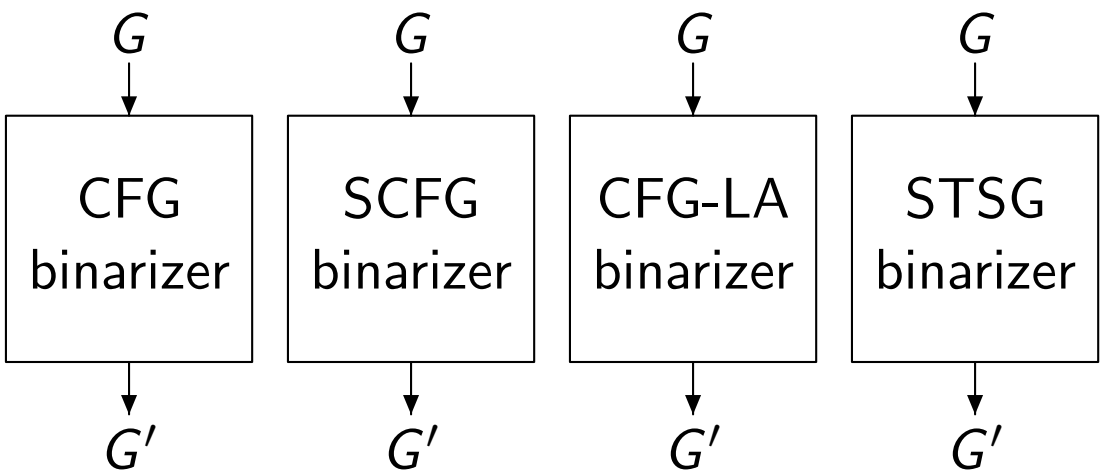
Further formalisms

- CFG (Chomsky)
- SCFG (Lewis and Stearns, 1966)
- CFG with latent annotations (Matsuzaki et al., 2005)



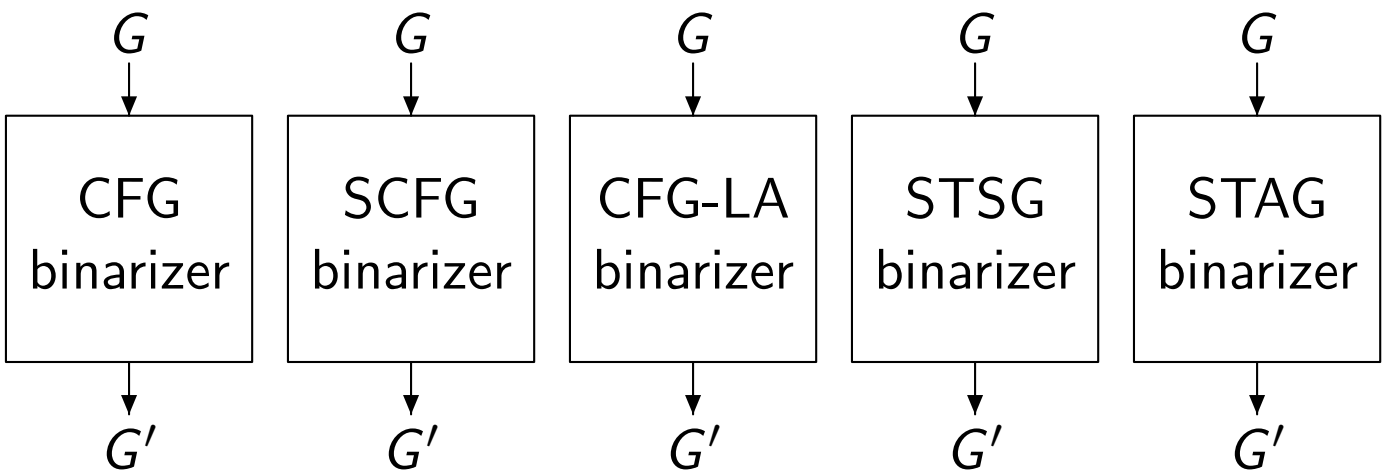
Further formalisms

- CFG (Chomsky)
- SCFG (Lewis and Stearns, 1966)
- CFG with latent annotations (Matsuzaki et al., 2005)
- synchronous tree-substitution grammars (Eisner, 2003)



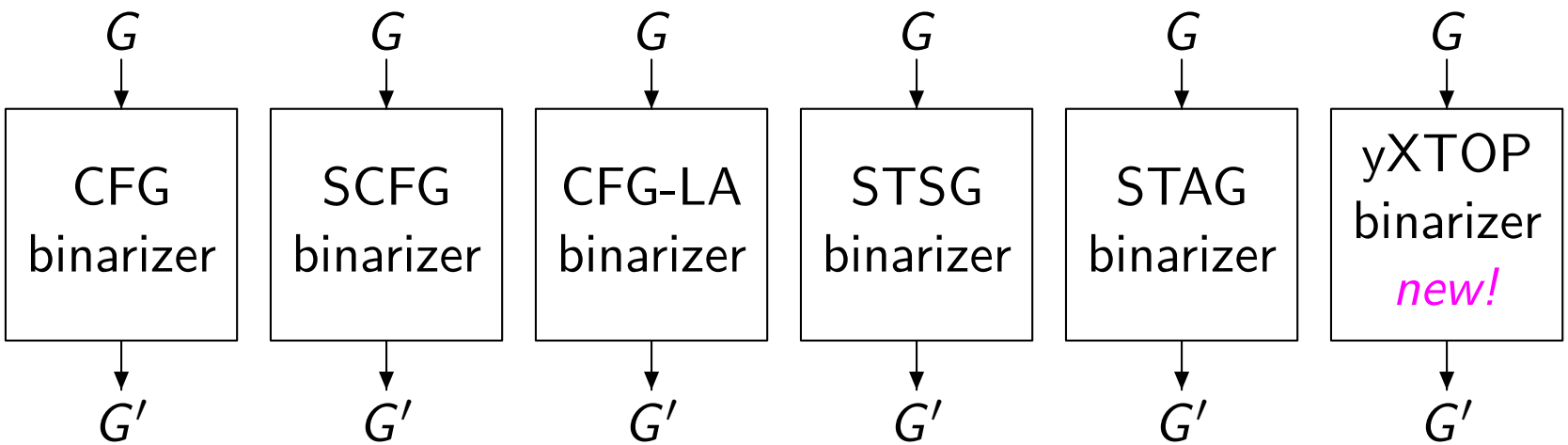
Further formalisms

- CFG (Chomsky)
- SCFG (Lewis and Stearns, 1966)
- CFG with latent annotations (Matsuzaki et al., 2005)
- synchronous tree-substitution grammars (Eisner, 2003)
- synchronous tree-adjoining grammars (Nesson et al., 2006)



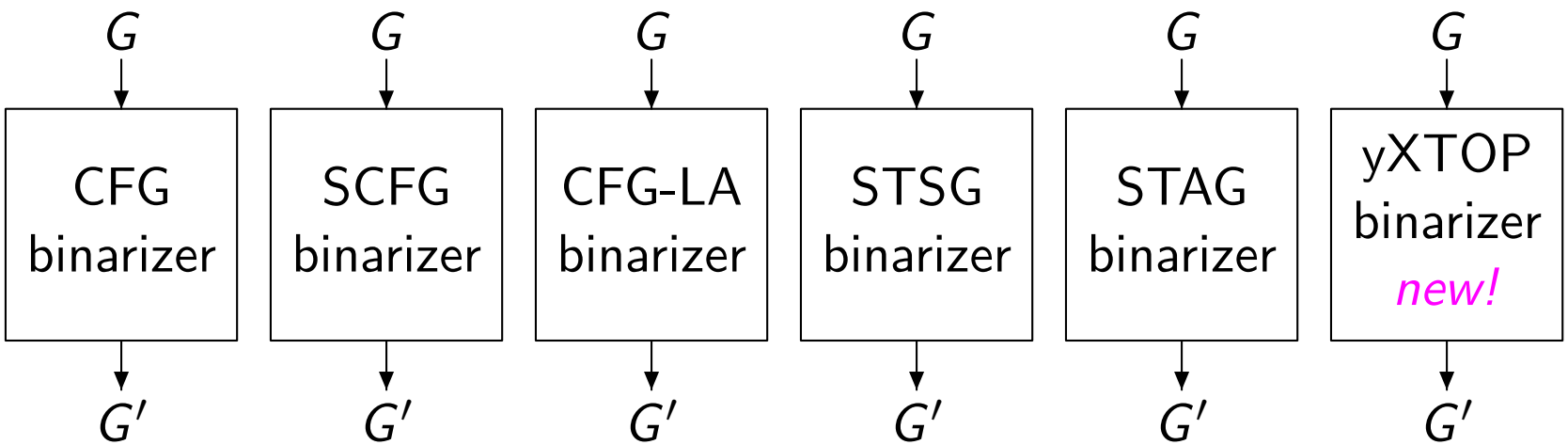
Further formalisms

- CFG (Chomsky)
- SCFG (Lewis and Stearns, 1966)
- CFG with latent annotations (Matsuzaki et al., 2005)
- synchronous tree-substitution grammars (Eisner, 2003)
- synchronous tree-adjoining grammars (Nesson et al., 2006)
- tree-to-string transducers (Graehl et al., 2008)



Further formalisms

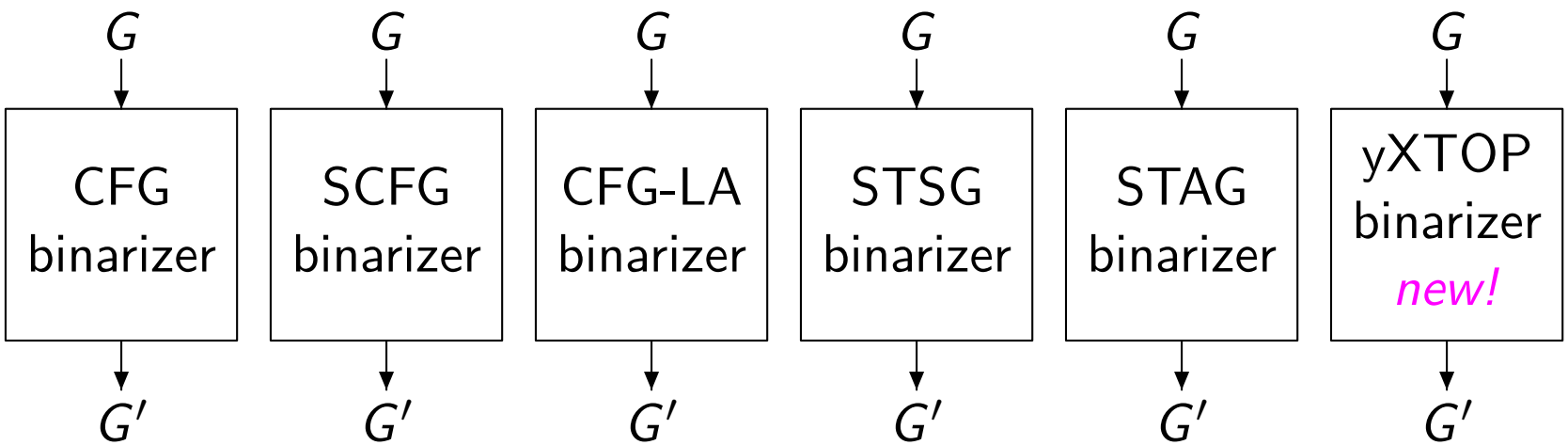
- CFG (Chomsky)
- SCFG (Lewis and Stearns, 1966)
- CFG with latent annotations (Matsuzaki et al., 2005)
- synchronous tree-substitution grammars (Eisner, 2003)
- synchronous tree-adjoining grammars (Nesson et al., 2006)
- tree-to-string transducers (Graehl et al., 2008)



six (potential) algorithms

Further formalisms

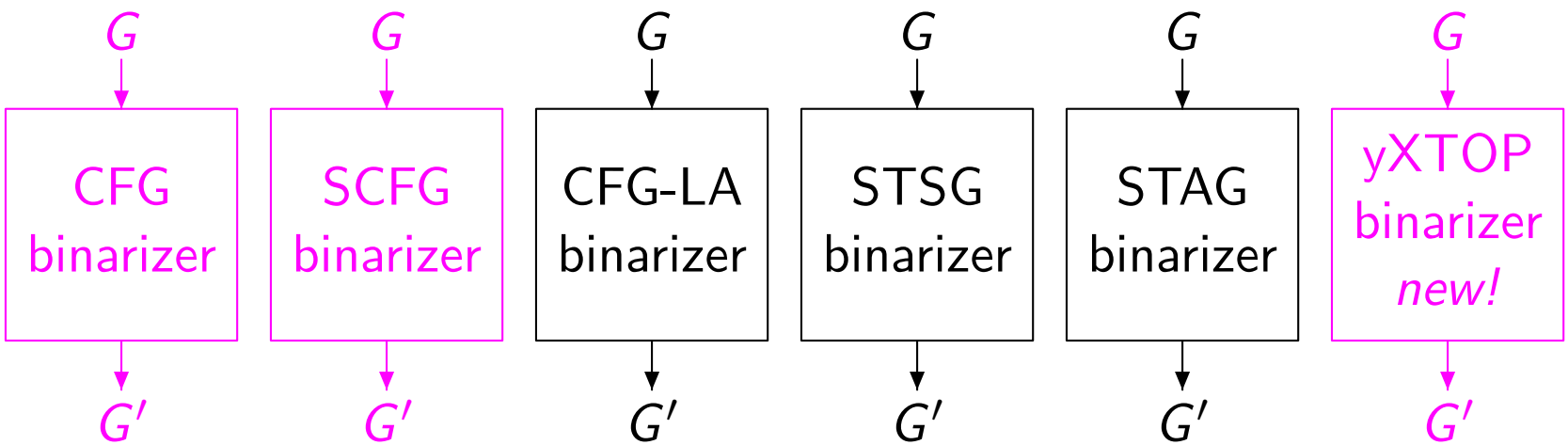
- CFG (Chomsky)
- SCFG (Lewis and Stearns, 1966)
- CFG with latent annotations (Matsuzaki et al., 2005)
- synchronous tree-substitution grammars (Eisner, 2003)
- synchronous tree-adjoining grammars (Nesson et al., 2006)
- tree-to-string transducers (Graehl et al., 2008)



six (potential) algorithms $\rightsquigarrow$ one shared algorithm!

Further formalisms

- CFG (Chomsky)
- SCFG (Lewis and Stearns, 1966)
- CFG with latent annotations (Matsuzaki et al., 2005)
- synchronous tree-substitution grammars (Eisner, 2003)
- synchronous tree-adjoining grammars (Nesson et al., 2006)
- tree-to-string transducers (Graehl et al., 2008)



six (potential) algorithms $\rightsquigarrow$ one shared algorithm!

Outline

Common data structure: IRTG

Every grammar formalism
mentioned so far
is a class of IRTGs.

IRTG . . . interpreted regular tree grammar

Example

synchronous context-free grammar:

$$A \rightarrow \langle BCD, DaBC \rangle$$
$$B \rightarrow \langle b, b \rangle$$
$$C \rightarrow \langle c, c \rangle$$
$$D \rightarrow \langle d, d \rangle$$

Example

synchronous context-free grammar:

$A \rightarrow \langle BCD, DaBC \rangle$
$B \rightarrow \langle b, b \rangle$
$C \rightarrow \langle c, c \rangle$
$D \rightarrow \langle d, d \rangle$

Example

synchronous context-free grammar:

$A \rightarrow \langle BCD, DaBC \rangle$
$B \rightarrow \langle b, b \rangle$
$C \rightarrow \langle c, c \rangle$
$D \rightarrow \langle d, d \rangle$

- derive pairs for B, C, and D

Example

synchronous context-free grammar:

$A \rightarrow \langle BCD, DaBC \rangle$
$B \rightarrow \langle b, b \rangle$
$C \rightarrow \langle c, c \rangle$
$D \rightarrow \langle d, d \rangle$

- derive pairs for B, C, and D
- construct
  component 1: $x_1\, x_2\, x_3$

Example

synchronous context-free grammar:

$A \rightarrow \langle BCD, DaBC \rangle$
$B \rightarrow \langle b, b \rangle$
$C \rightarrow \langle c, c \rangle$
$D \rightarrow \langle d, d \rangle$

- derive pairs for B, C, and D
- construct
  - component 1: $x_1\, x_2\, x_3$
  - component 2: $x_3\, a\, x_1\, x_2$

Example

synchronous context-free grammar:

$A \to \langle BCD, DaBC \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

- derive pairs for B, C, and D
- construct

  component 1: $x_1\, x_2\, x_3$
  component 2: $x_3\, a\, x_1\, x_2$

  } α

Example

synchronous context-free grammar:

$A \to \langle BCD, DaBC \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

- derive pairs for B, C, and D
- construct

$$\left.\begin{array}{ll} \text{component 1:} & x_1\, x_2\, x_3 \\ \text{component 2:} & x_3\, a\, x_1\, x_2 \end{array}\right\} \alpha$$

interpreted regular tree grammar:

$$\begin{array}{lccc} & & h_1 & h_2 \\ A \to \alpha(B, C, D) & \mathrm{con}^3(x_1, x_2, x_3) & \longleftarrow \alpha & \longmapsto \mathrm{con}^4(x_3, a, x_1, x_2) \\ B \to \alpha_1 & b & \longleftarrow \alpha_1 & \longmapsto b \\ C \to \alpha_2 & c & \longleftarrow \alpha_2 & \longmapsto c \\ D \to \alpha_3 & d & \longleftarrow \alpha_3 & \longmapsto d \end{array}$$

Example

synchronous context-free grammar:

$A \to \langle BCD, DaBC \rangle$
$B \to \langle b, b \rangle$
$C \to \langle c, c \rangle$
$D \to \langle d, d \rangle$

- derive pairs for B, C, and D
- construct

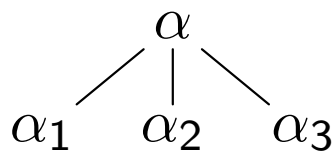
$$\left.\begin{array}{ll} \text{component 1:} & x_1\, x_2\, x_3 \\ \text{component 2:} & x_3\, a\, x_1\, x_2 \end{array}\right\} \alpha$$

interpreted regular tree grammar:

$$\begin{array}{lccc} & & h_1 & h_2 \\ A \to \alpha(B, C, D) & \mathrm{con}^3(x_1, x_2, x_3) & \longleftarrow \alpha & \longmapsto \mathrm{con}^4(x_3, a, x_1, x_2) \\ B \to \alpha_1 & & b \longleftarrow \alpha_1 & \longmapsto b \\ C \to \alpha_2 & & c \longleftarrow \alpha_2 & \longmapsto c \\ D \to \alpha_3 & & d \longleftarrow \alpha_3 & \longmapsto d \end{array}$$

Semantics

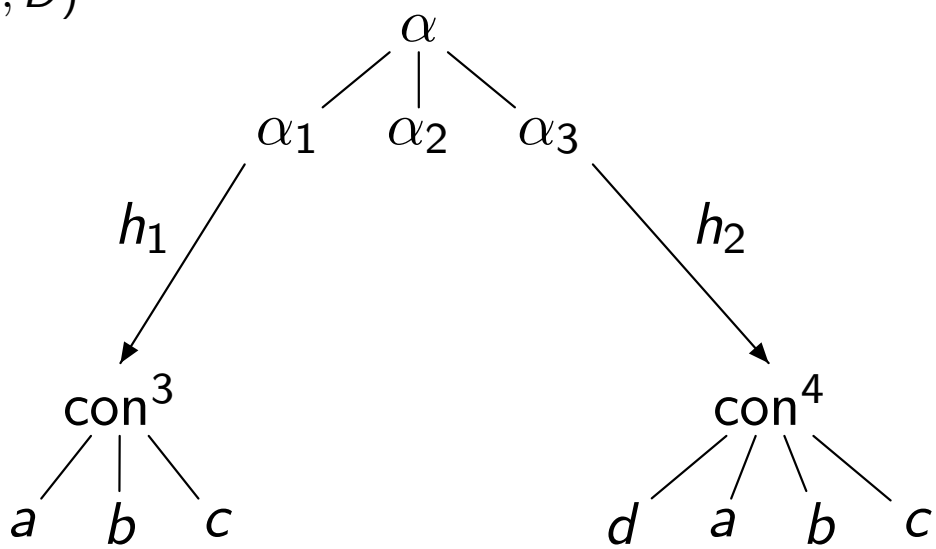
$A \to \alpha(B, C, D)$
$B \to \alpha_1$
$C \to \alpha_2$
$D \to \alpha_3$



derivation tree

Semantics

$A \to \alpha(B, C, D)$
$B \to \alpha_1$
$C \to \alpha_2$
$D \to \alpha_3$

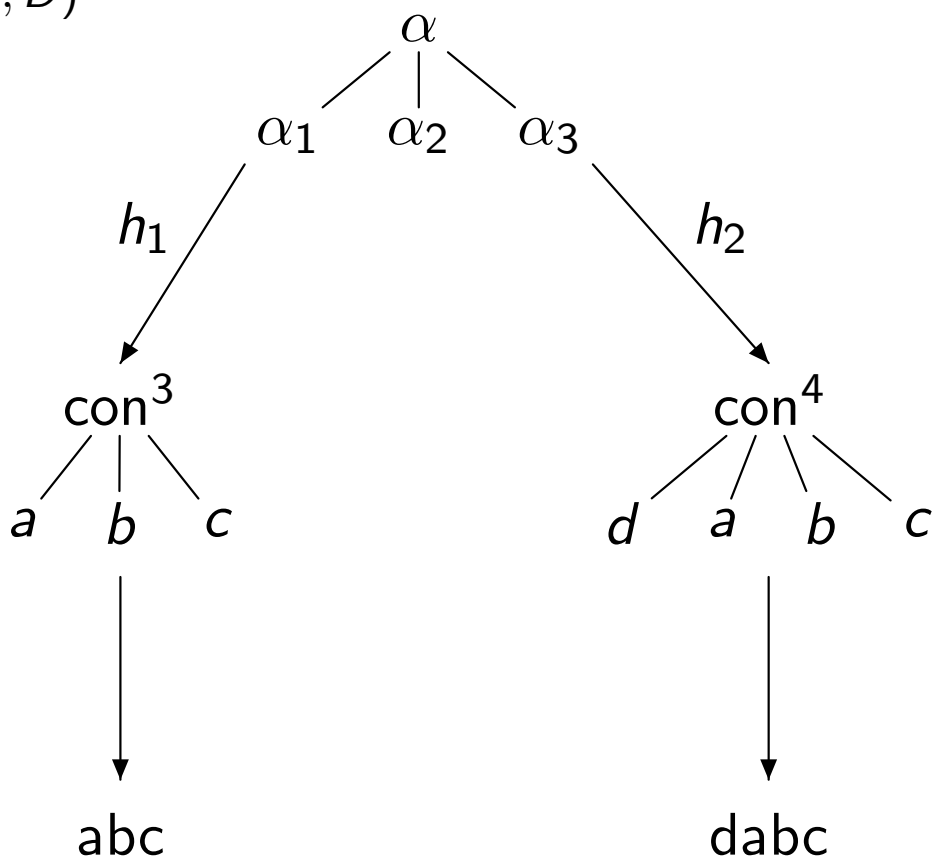


derivation tree

semantic term

Semantics

$A \to \alpha(B, C, D)$
$B \to \alpha_1$
$C \to \alpha_2$
$D \to \alpha_3$



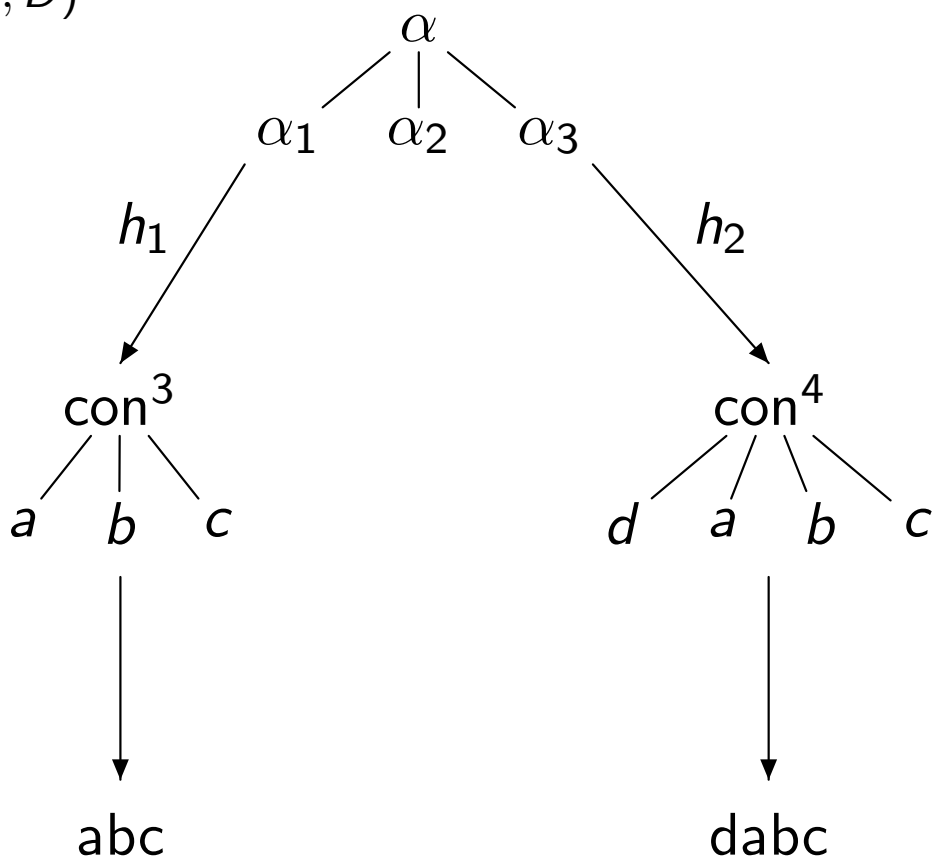
derivation tree

semantic term

semantic object

Semantics

$A \to \alpha(B, C, D)$
$B \to \alpha_1$
$C \to \alpha_2$
$D \to \alpha_3$



derivation tree

semantic term

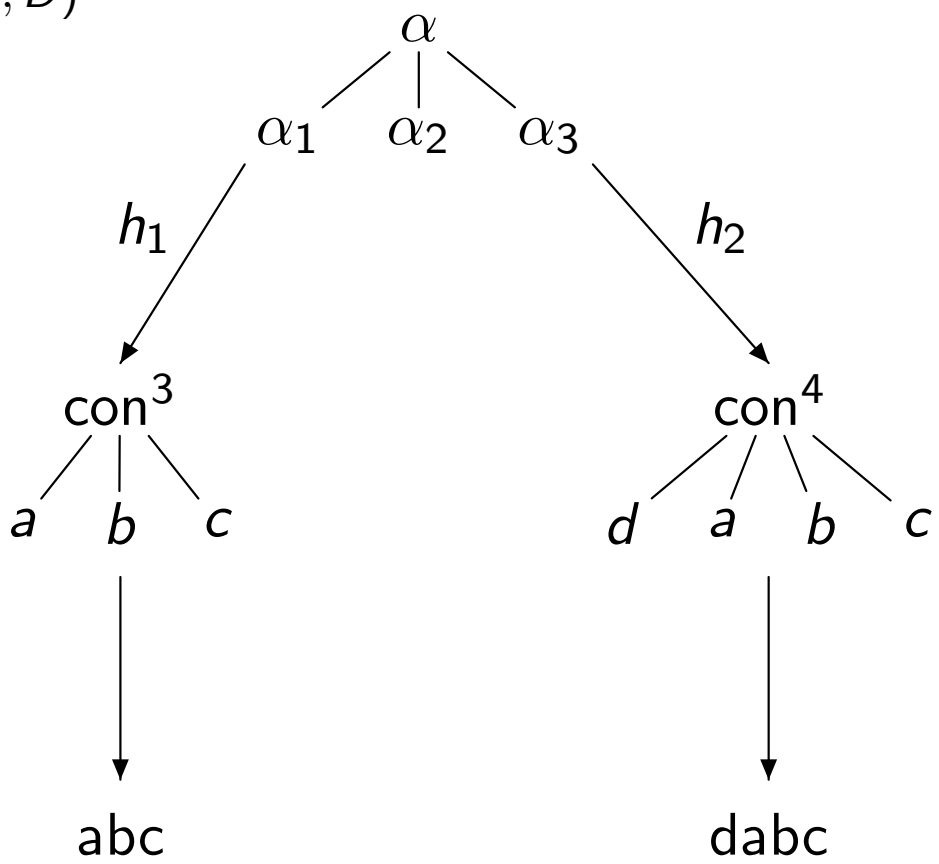
semantic object

interpretation 1

interpretation 2

Semantics

$A \to \alpha(B, C, D)$
$B \to \alpha_1$
$C \to \alpha_2$
$D \to \alpha_3$



derivation tree

semantic term

semantic object

interpretation 1

interpretation 2

n . . . number of interpretations

Why the trouble?

Why the trouble?

Every grammar formalism
mentioned so far
is a class of IRTGs.

Outline

Binarization problem

rule-by-rule binarization problem *given* G:

for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

Binarization problem

rule-by-rule binarization problem *given* G:

for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

binarization problem:

convert given IRTG G into equivalent IRTG G' of rank 2

Binarization problem

rule-by-rule binarization problem *given* G:

for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

binarization problem:

convert given IRTG G into equivalent IRTG G' of rank 2

Binarization problem

rule-by-rule binarization problem *given* G:

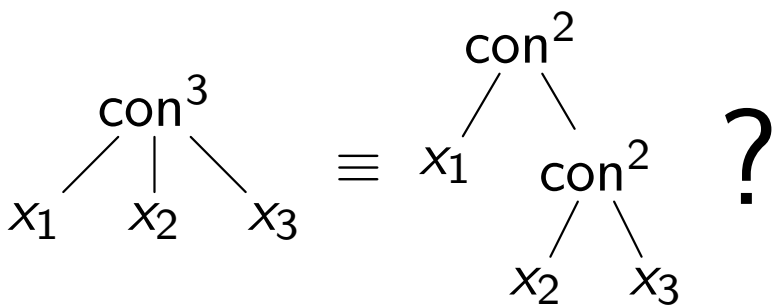
for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

binarization problem:

convert given IRTG G into equivalent IRTG G' of rank 2

equivalence problem: is it true that



Binarization problem

rule-by-rule binarization problem *given* G:

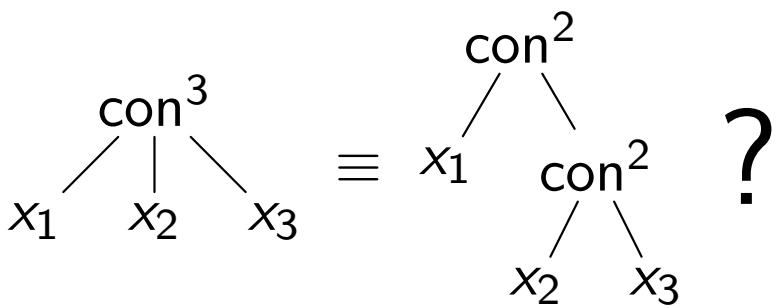
for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

binarization problem:

convert given IRTG G into equivalent IRTG G' of rank 2

equivalence problem: is it true that



undecidable in general

Binarization problem

rule-by-rule binarization problem *given* G:

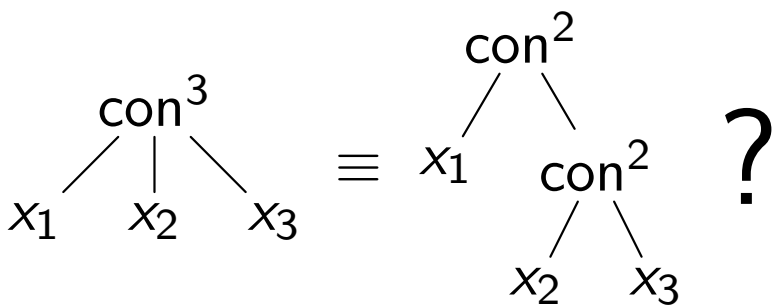
for each rule of rank > 2

find equivalent collection of rules of rank ≤ 2

binarization problem:

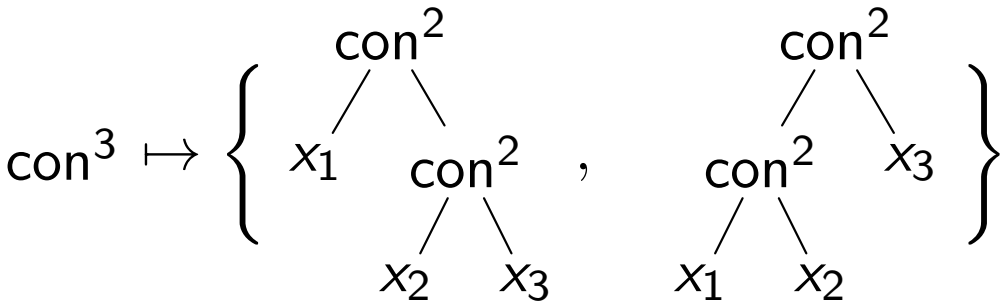
convert given IRTG G into equivalent IRTG G' of rank 2

equivalence problem: is it true that

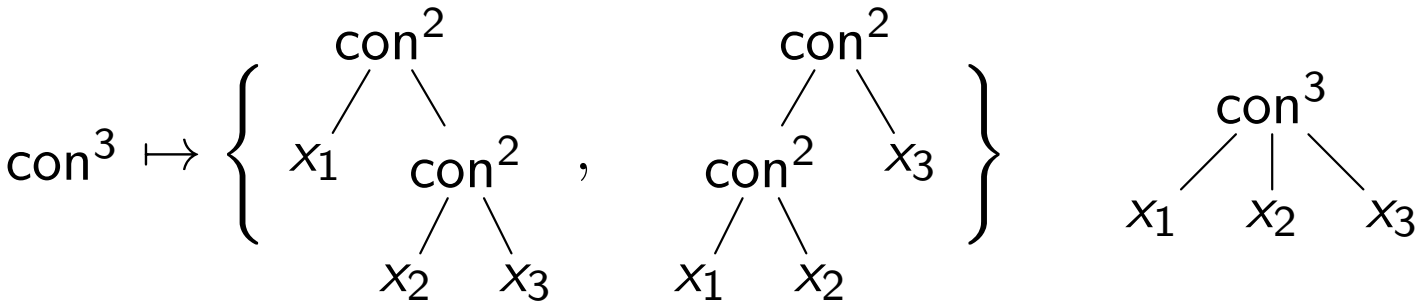


undecidable in general $\rightsquigarrow$ adapt problem specification!

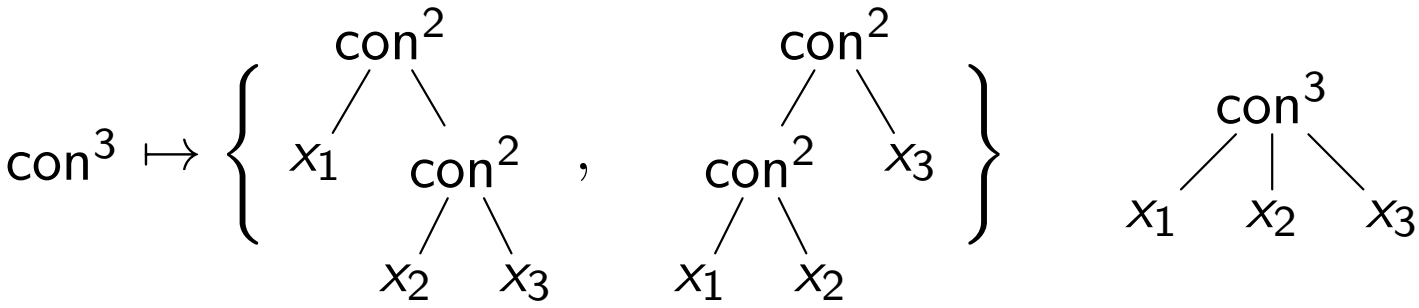
Binarization rule $\mathfrak{b}$: operation symbols $\to$ binary terms



Binarization rule $\mathfrak{b}$: operation symbols $\to$ binary terms

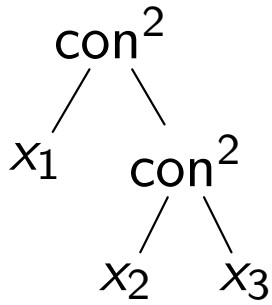
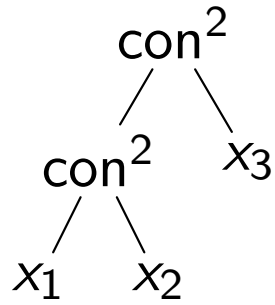


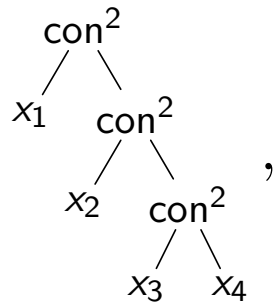
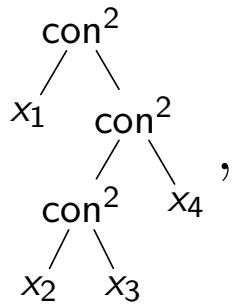
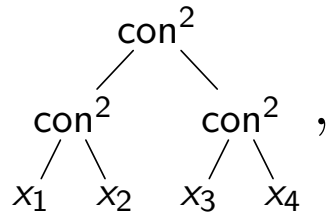
Binarization rule $\flat$: operation symbols $\to$ binary terms

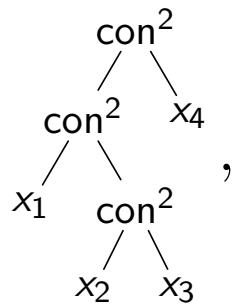
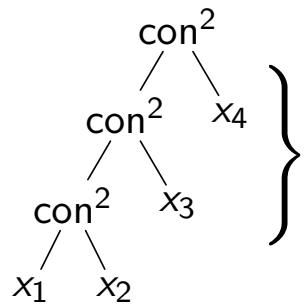


$$x_1 \cdot (x_2 \cdot x_3) = (x_1 \cdot x_2) \cdot x_3 \quad = \quad x_1 \cdot x_2 \cdot x_3$$

Binarization rule $\mathfrak{b}$: operation symbols $\to$ binary terms

$\mathrm{con}^3 \mapsto \left\{ \right.$  ,  $\left. \right\}$

$\mathrm{con}^4 \mapsto \left\{ \right.$  ,  ,  ,

 ,  $\left. \right\}$

$\vdots$

Refined binarization problem

rule-by-rule binarization problem *given* G *with respect to* $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$:

for each rule of rank > 2

find collection of rules of rank ≤ 2

equivalent with respect to $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$

Refined binarization problem

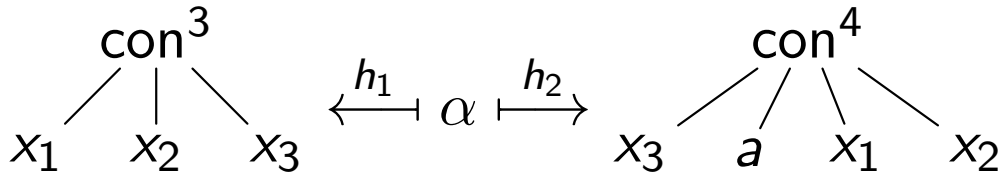
rule-by-rule binarization problem *given* G *with respect to* $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$:

for each rule of rank > 2
find collection of rules of rank ≤ 2
equivalent with respect to $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$

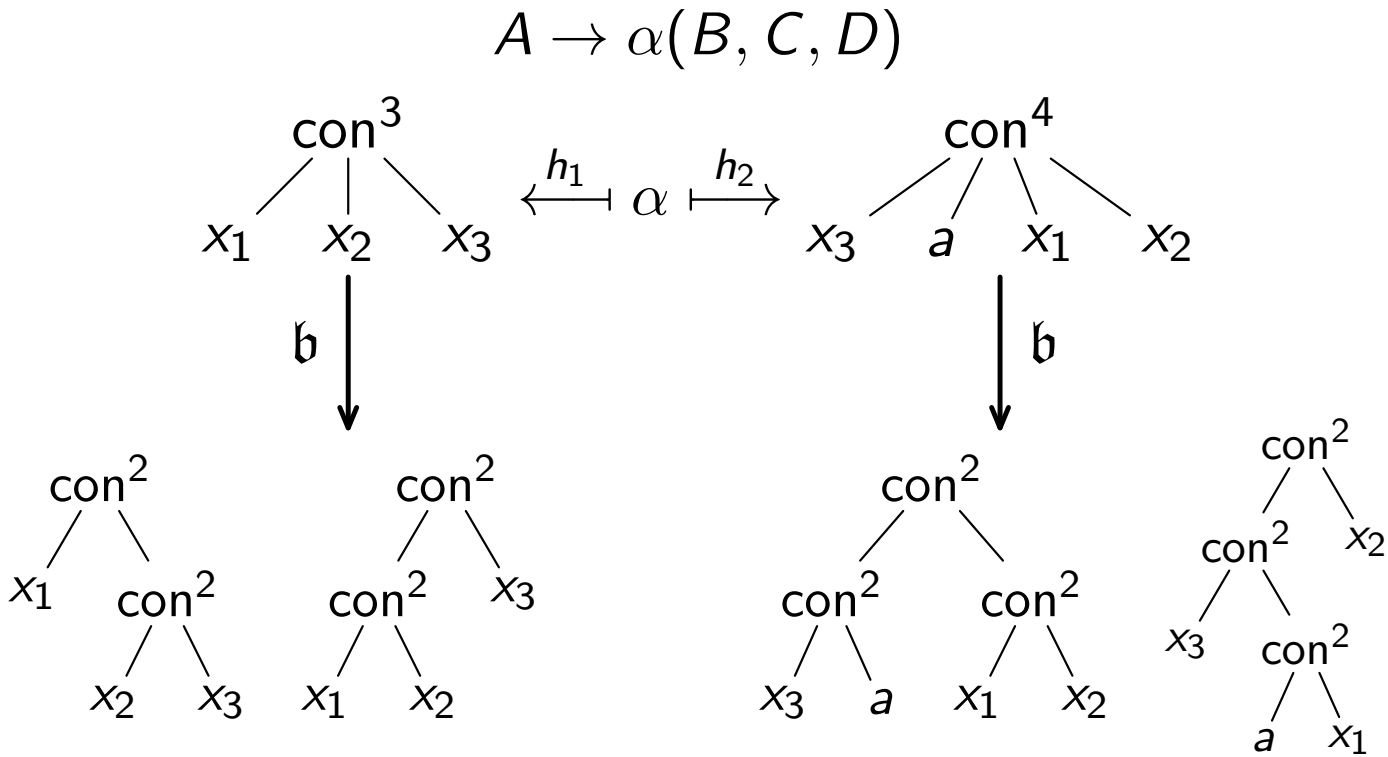
remember: solution need not exist

Using the binarization rule $\flat$ constructively

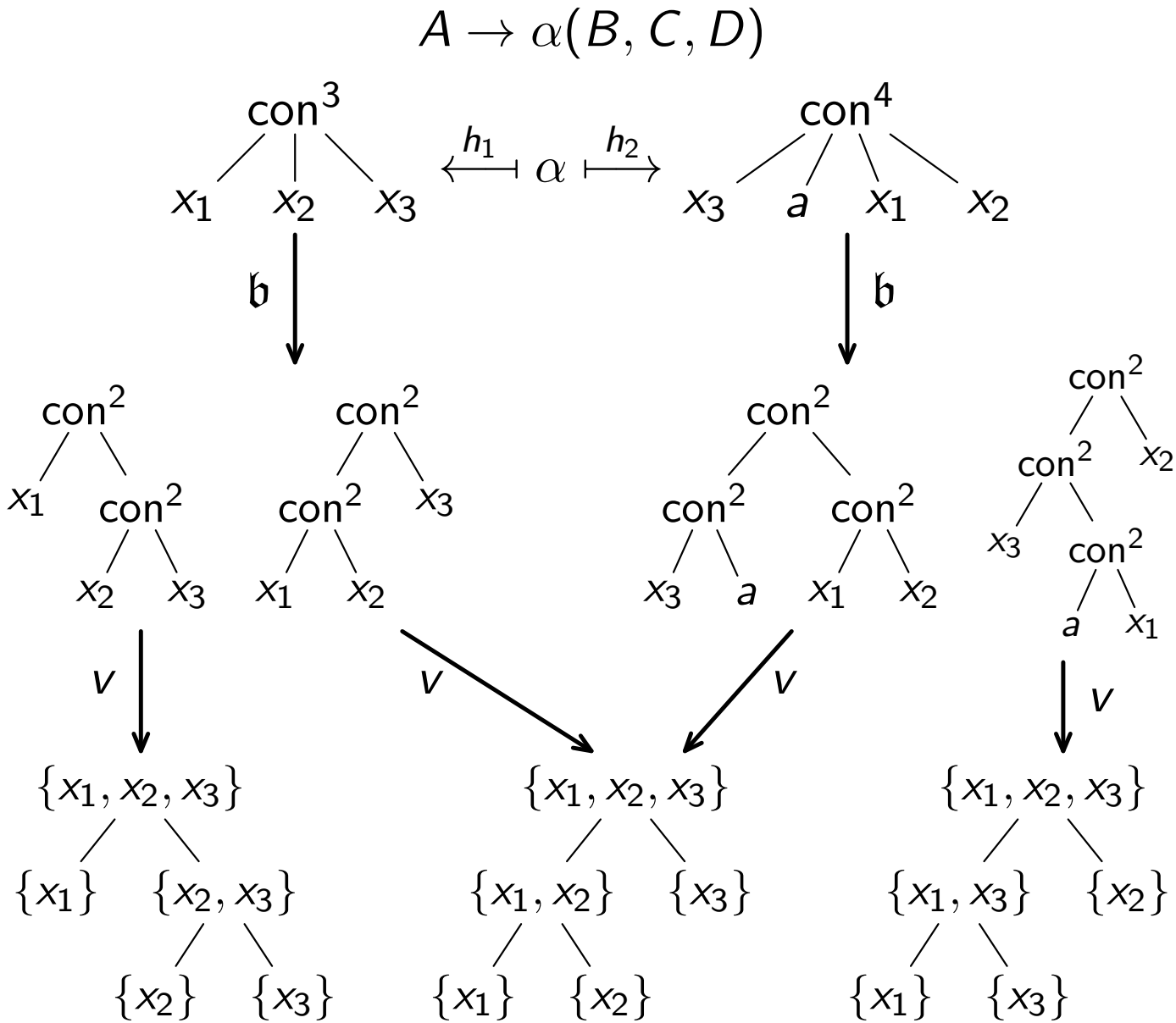
$$A \to \alpha(B, C, D)$$



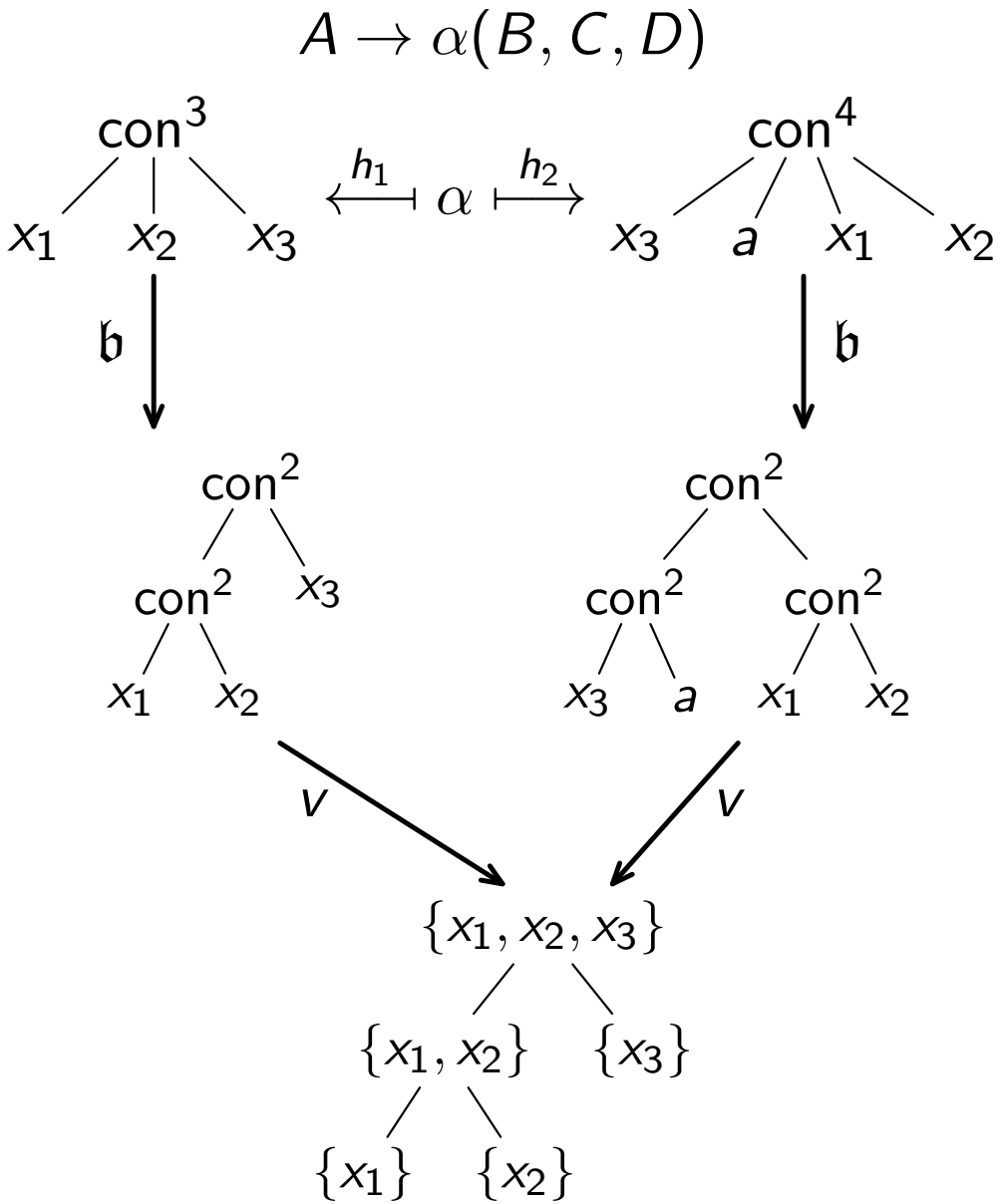
Using the binarization rule $\mathfrak{b}$ constructively



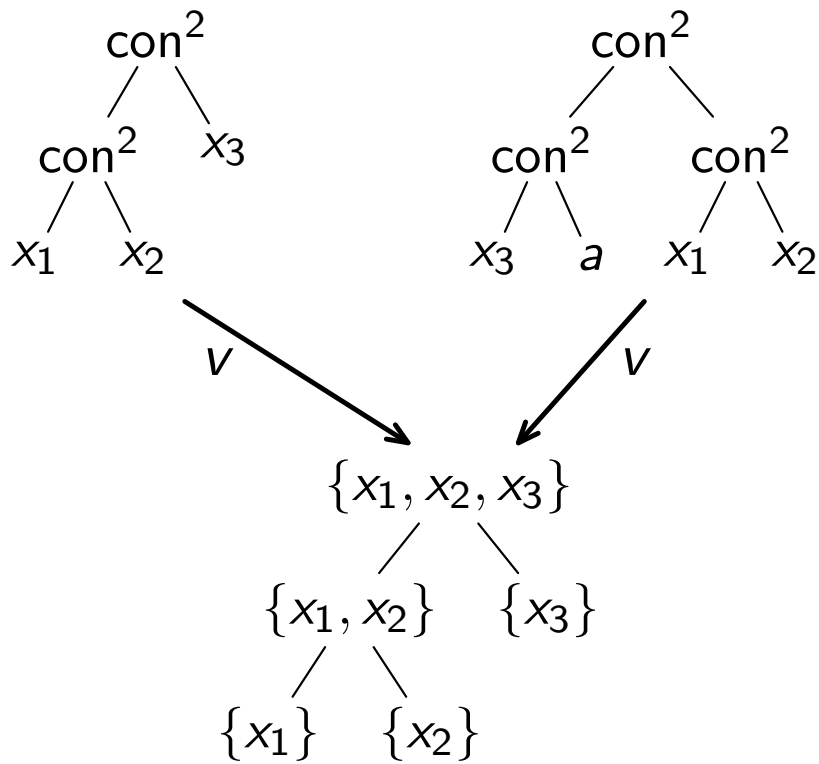
Using the binarization rule $\mathfrak{b}$ constructively



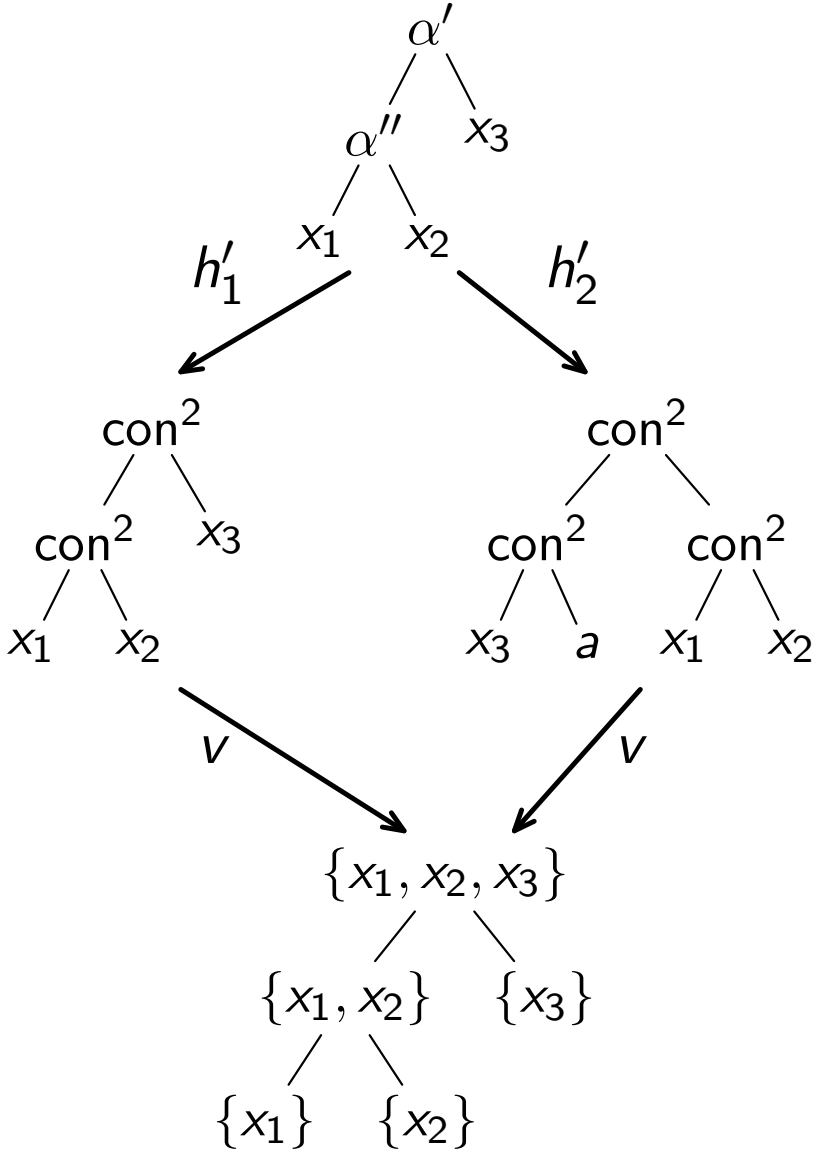
Using the binarization rule $\mathfrak{b}$ constructively



Using the binarization rule $\flat$ constructively

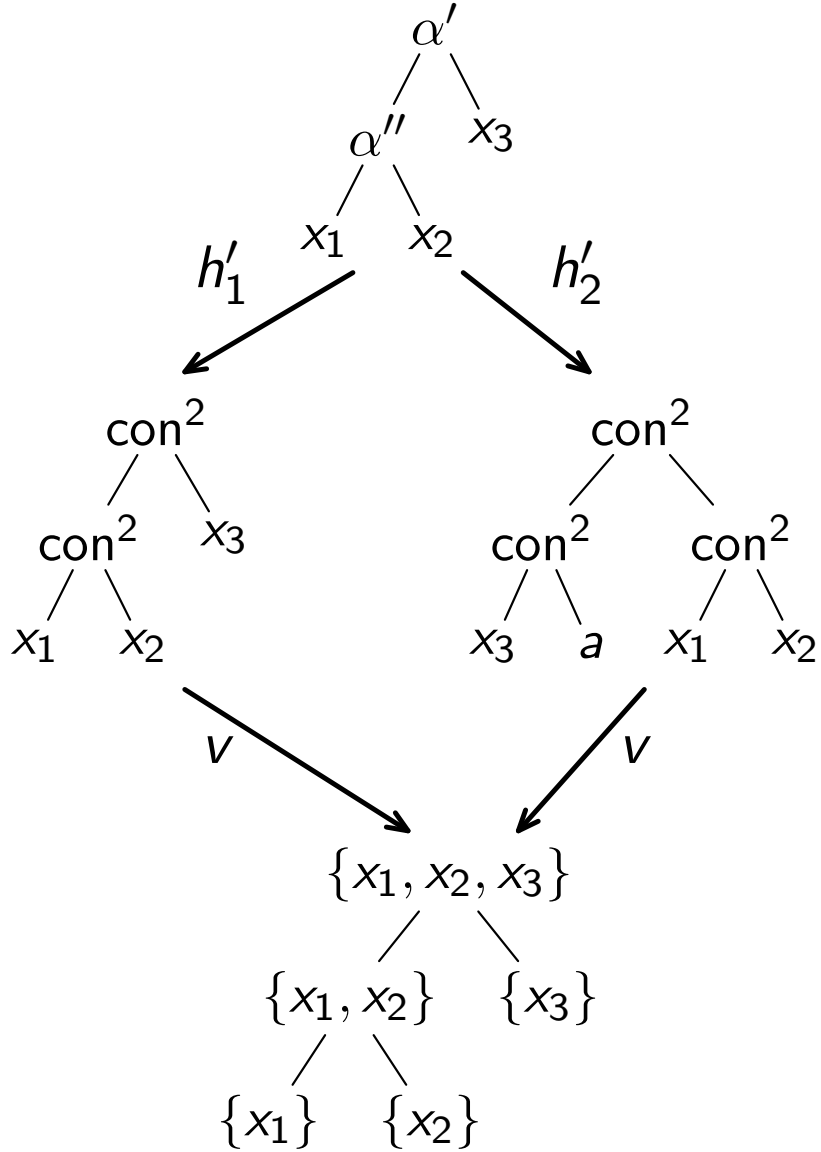


Using the binarization rule $\flat$ constructively



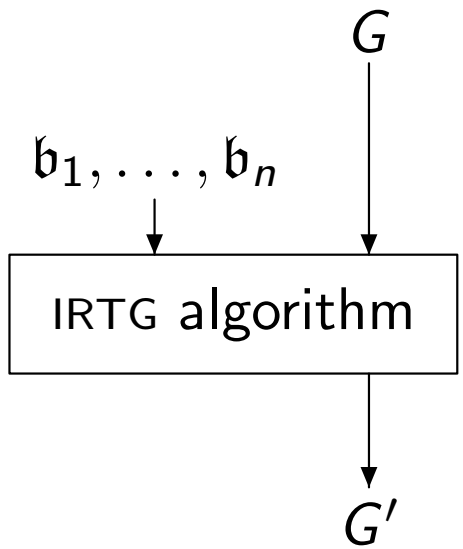
Using the binarization rule $\flat$ constructively

$$A \to \alpha'([BC], D) \qquad [BC] \to \alpha''(B, C)$$

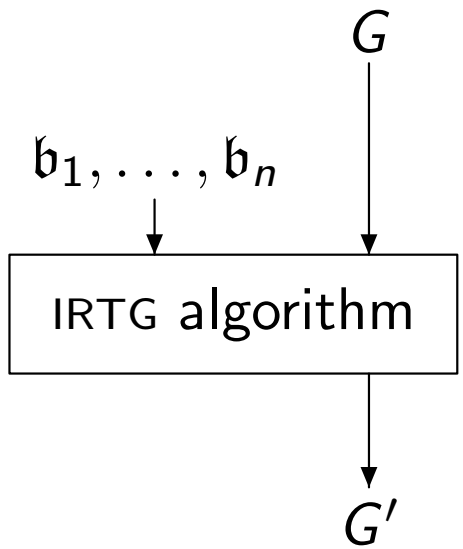


Outline

Algorithm

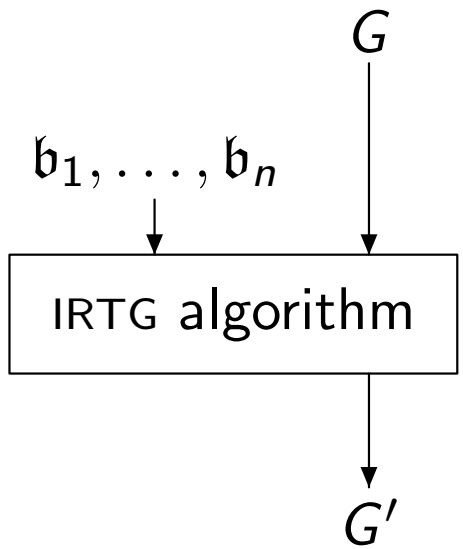


Algorithm



- G' is equivalent to G

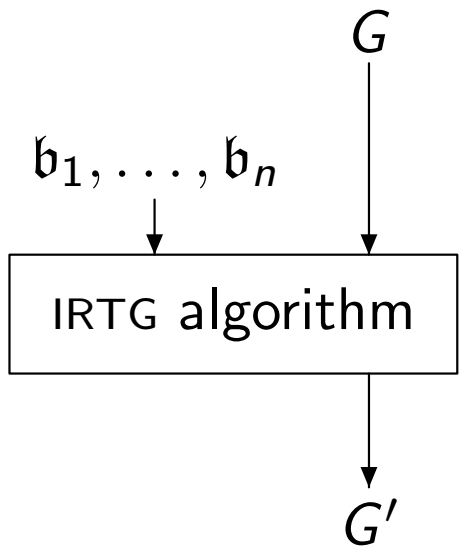
Algorithm



- G' is equivalent to G
- G' has rank 2

  if the rule-by-rule binarization problem *given* G with respect to $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$ has a solution

Algorithm

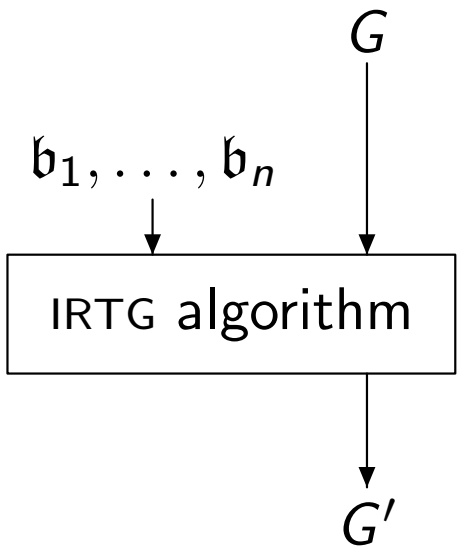


- G' is equivalent to G
- G' has rank 2

  if the rule-by-rule binarization problem *given* G with respect to $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$ has a solution

  otherwise: best effort

Algorithm

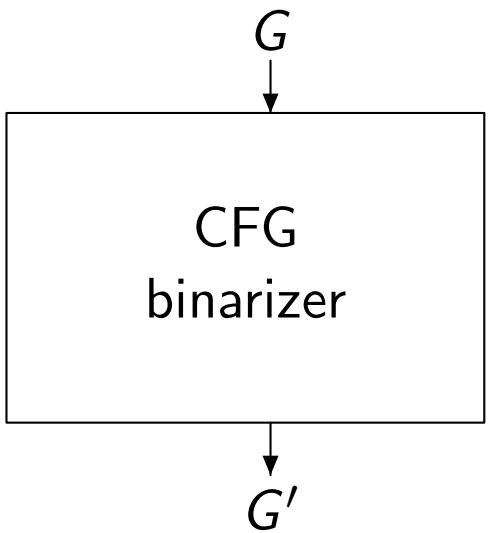


- G' is equivalent to G
- G' has rank 2
  if the rule-by-rule binarization problem *given* G with respect to $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$ has a solution
  otherwise: best effort
- runtime $O(|R| \cdot c^n)$

Applications

Solve rule-by-rule binarization problem (whenever possible)

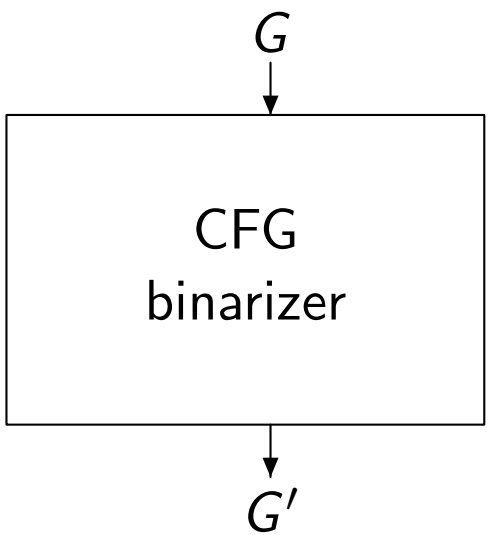
for every CFG:



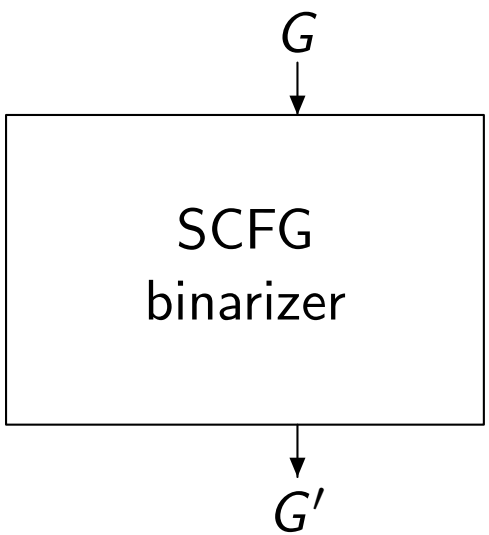
Applications

Solve rule-by-rule binarization problem (whenever possible)

for every CFG:



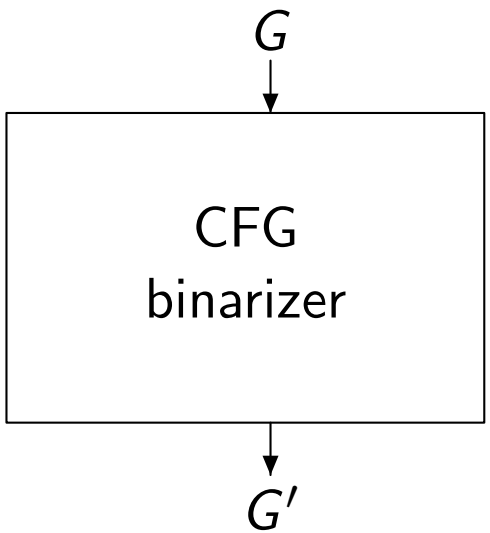
for every SCFG:



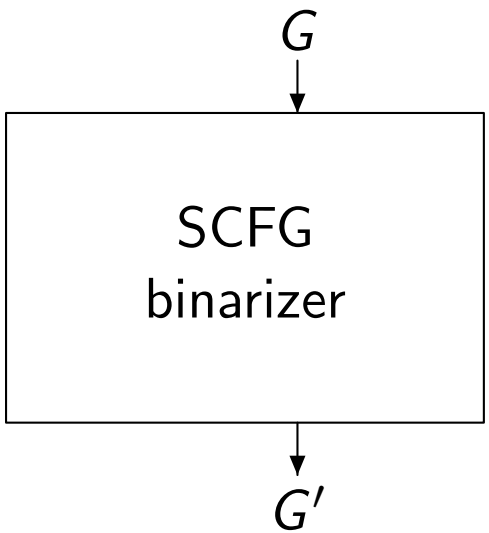
Applications

Solve rule-by-rule binarization problem (whenever possible)

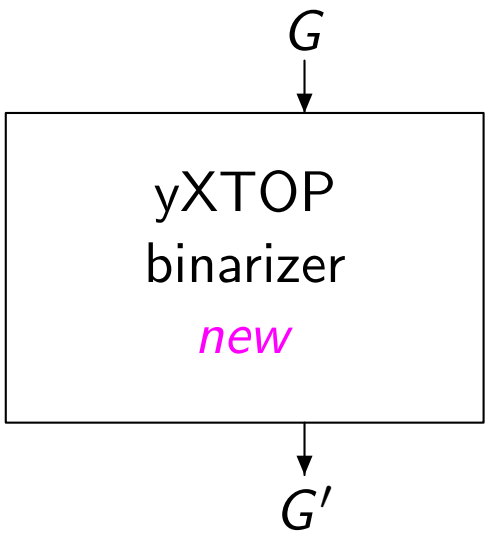
for every CFG:



for every SCFG:



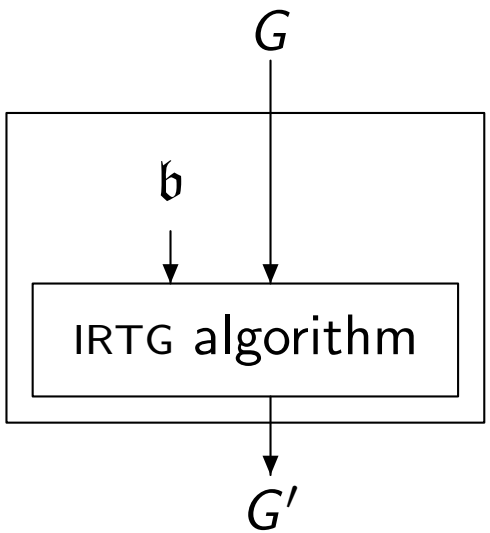
for every
tree-to-string
transducer:



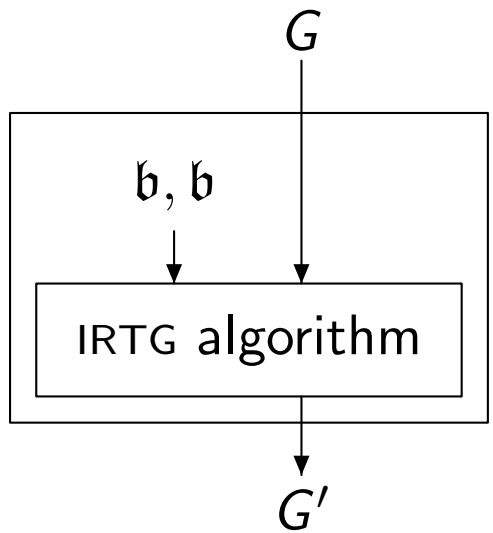
Applications

Solve rule-by-rule binarization problem (whenever possible)

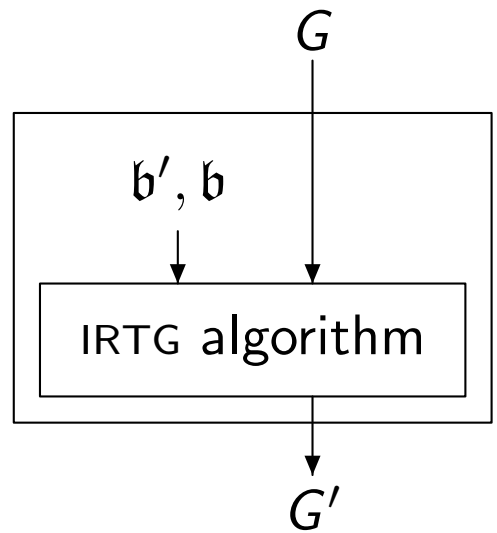
for every CFG:



for every SCFG:



for every tree-to-string transducer:

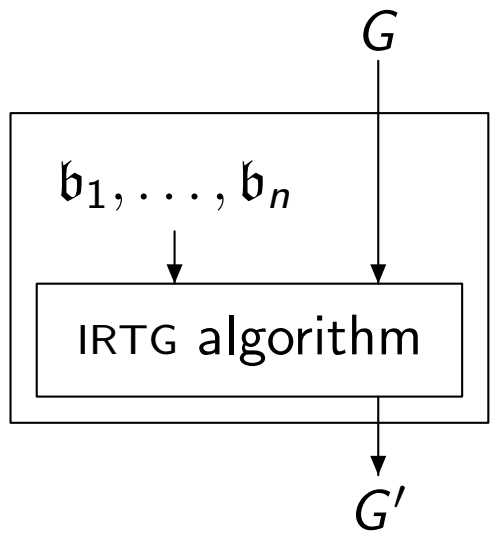


. . . with merely two binarization rules $\mathfrak{b}, \mathfrak{b}'$!

More generally

Solve rule-by-rule binarization problem (whenever possible)

for every $G \in \mathcal{C}$:



provided that $\mathfrak{b}_1, \ldots, \mathfrak{b}_n$ are complete for every $G \in \mathcal{C}$

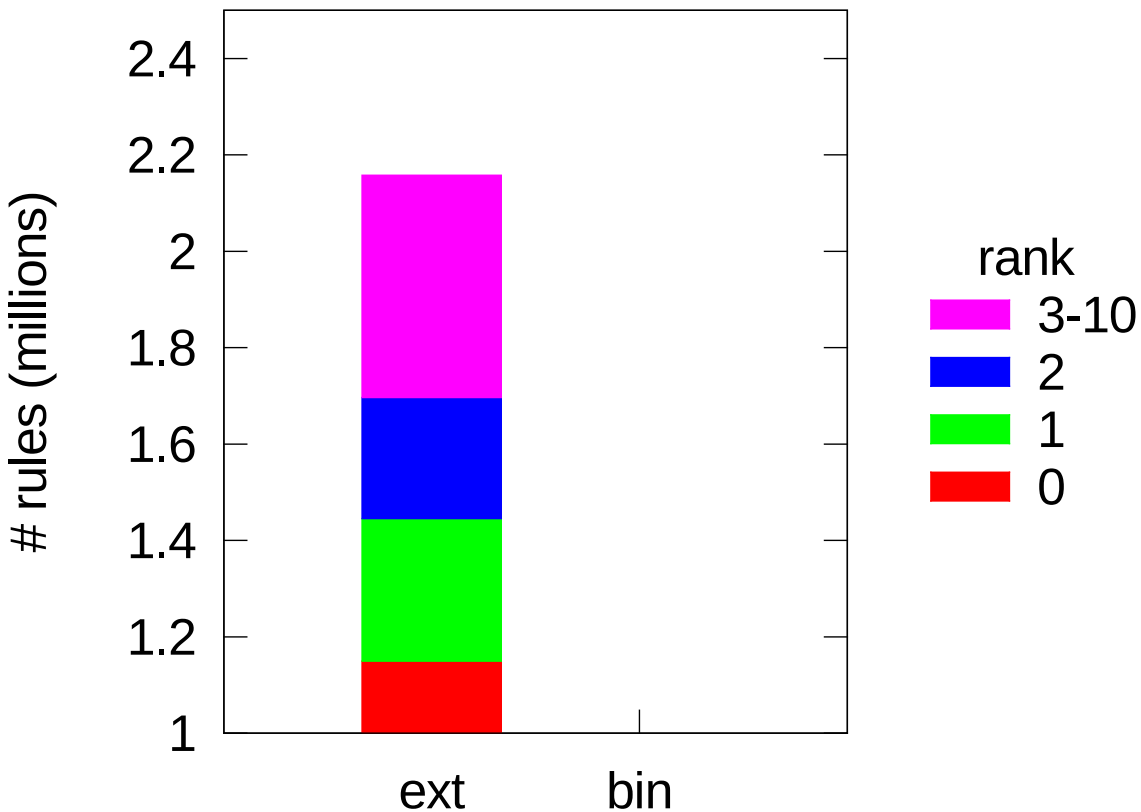
Outline

Experiment: tree-to-string transducers

- extracted from 1 M parallel sentences (Europarl Eng-Ger)

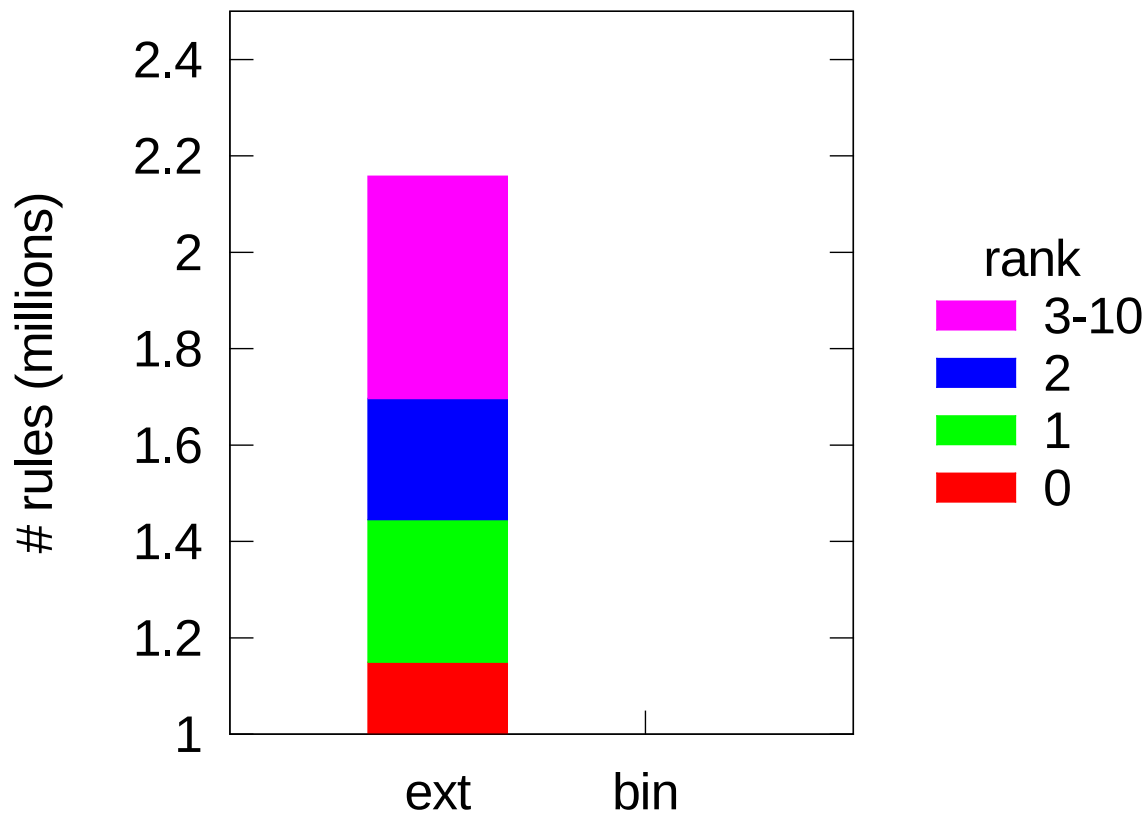
Experiment: tree-to-string transducers

- extracted from 1 M parallel sentences (Europarl Eng-Ger)
- 2.15 M rules up to rank 10



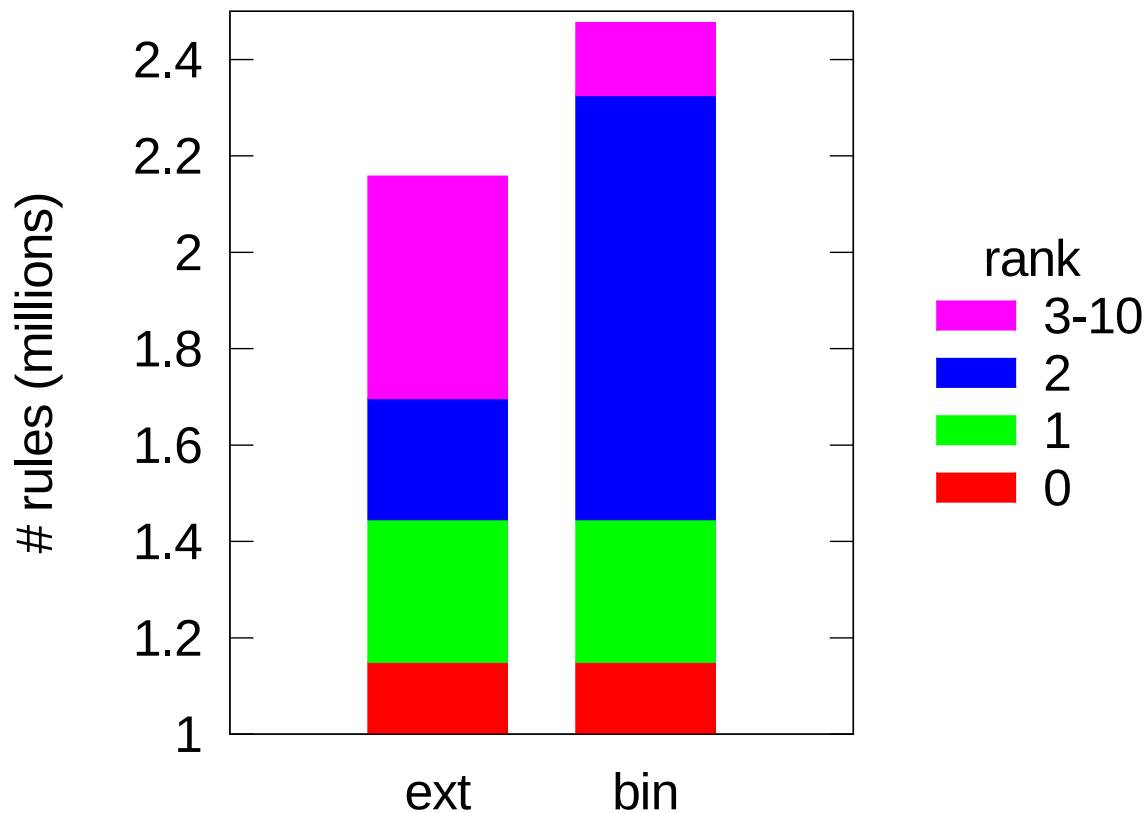
Experiment: tree-to-string transducers

- extracted from 1 M parallel sentences (Europarl Eng-Ger)
- 2.15 M rules up to rank 10
- implementation in Haskell



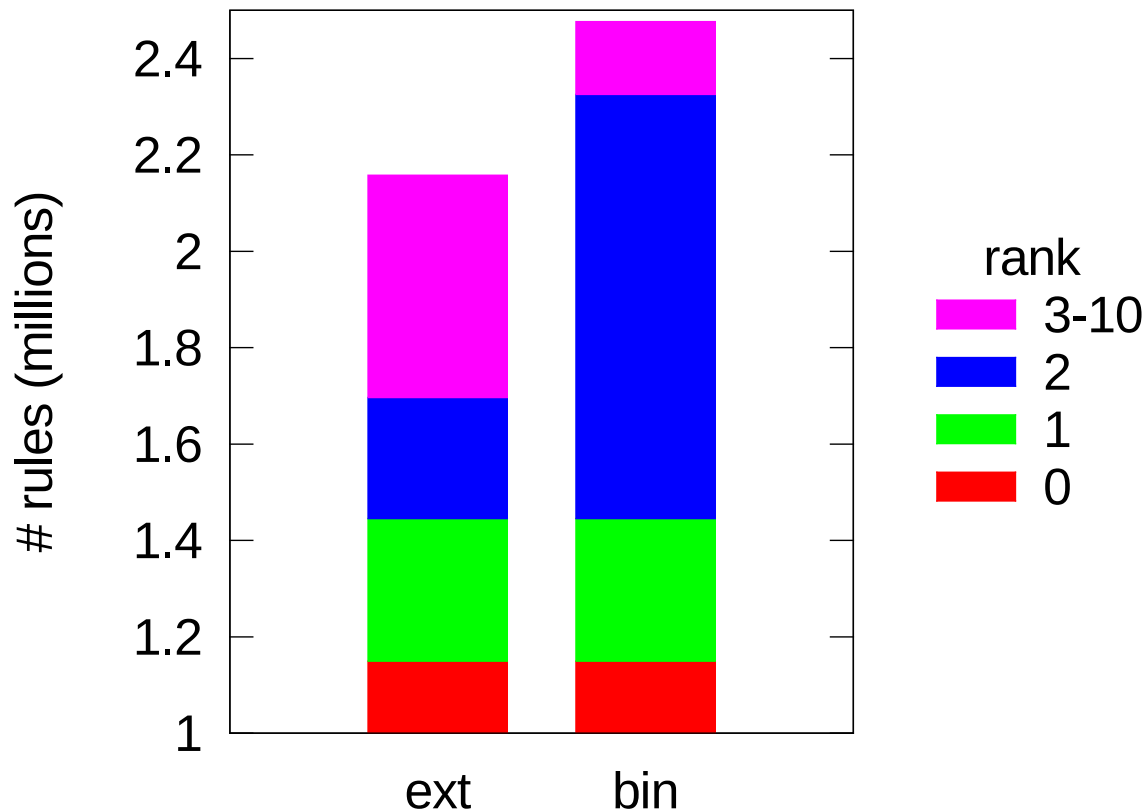
Experiment: tree-to-string transducers

- extracted from 1 M parallel sentences (Europarl Eng-Ger)
- 2.15 M rules up to rank 10
- implementation in Haskell
- 67% rules could be binarized



Experiment: tree-to-string transducers

- extracted from 1 M parallel sentences (Europarl Eng-Ger)
- 2.15 M rules up to rank 10
- implementation in Haskell
- 67% rules could be binarized
- took 4.4 min on a single core of an i5 2520M



Conclusion

Need to binarize?
The algorithm is there.
Supply binarization rules.

Thank you!