# Lexicalized Parsing

**Christoph Dörr**

# Übersicht

- Probabilistic CFG
- Bilexical CFG
- Parsen von bilexikalisierten CFG in $O(n^4)$
- Head Automaton Grammars und Split Grammars
- Parsen von bilexikalisierten CFG in $O(n^3)$ (für den Spezialfall Split Grammars)

# Probabilistic CFG

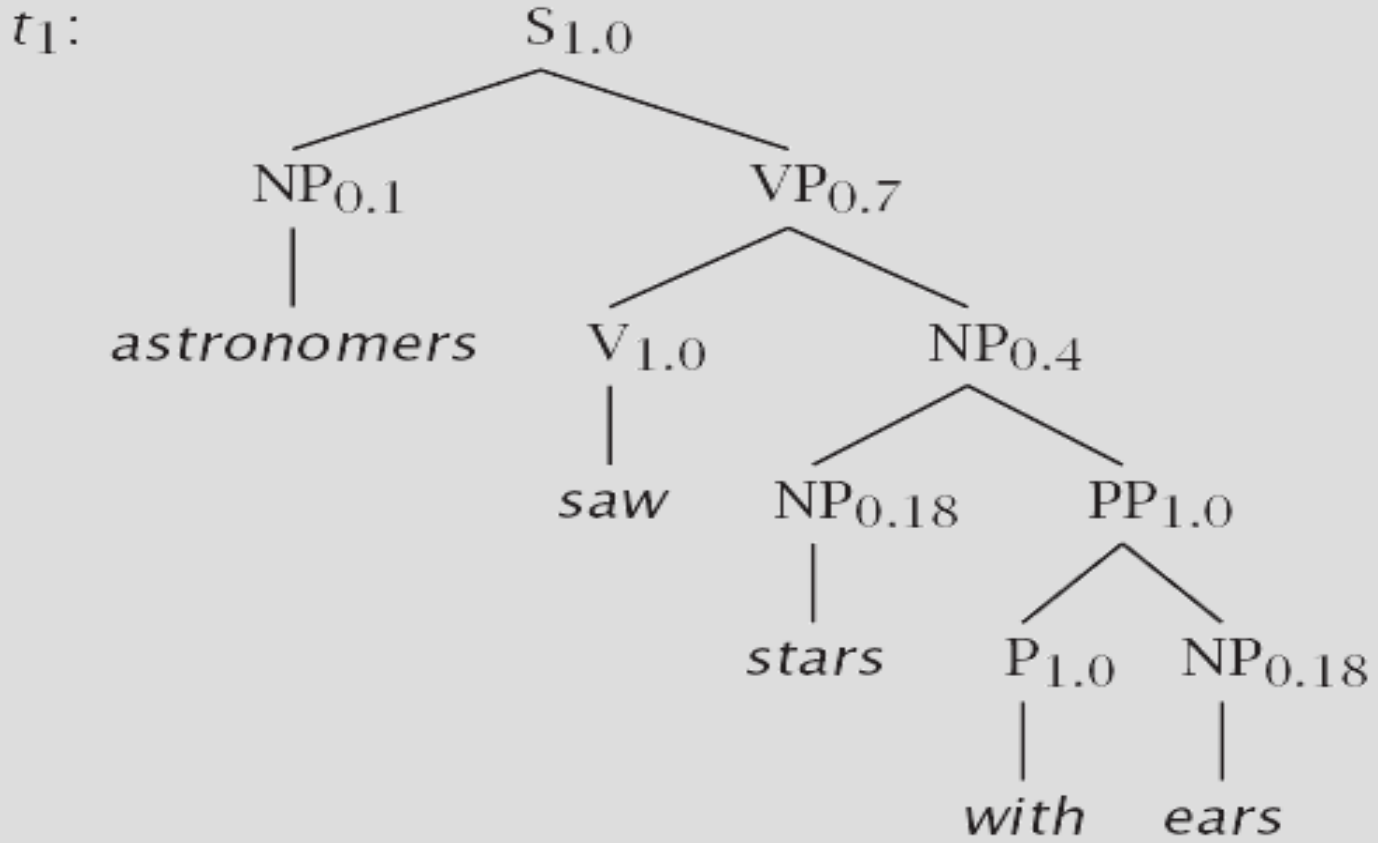
- Ordnet jeder Ableitungsregel eine Wahrscheinlichkeit zu, so dass die Summe der Wahrscheinlichkeiten aller Ableitungen für jedes Nichtterminal 1 ist.

- Die Wahrscheinlichkeit einer bestimmten Ableitung $ =>* w  t ist gleich dem Produkt der Wahrscheinlichkeiten der verwendeten Ableitungsregeln.

# Beispiel PCFG

| | | | |
|---|---|---|---|
| S → NP VP | 1.0 | NP → NP PP | 0.4 |
| PP → P NP | 1.0 | NP → *astronomers* | 0.1 |
| VP → V NP | 0.7 | NP → *ears* | 0.18 |
| VP → VP PP | 0.3 | NP → *saw* | 0.04 |
| P → *with* | 1.0 | NP → *stars* | 0.18 |
| V → *saw* | 1.0 | NP → *telescopes* | 0.1 |

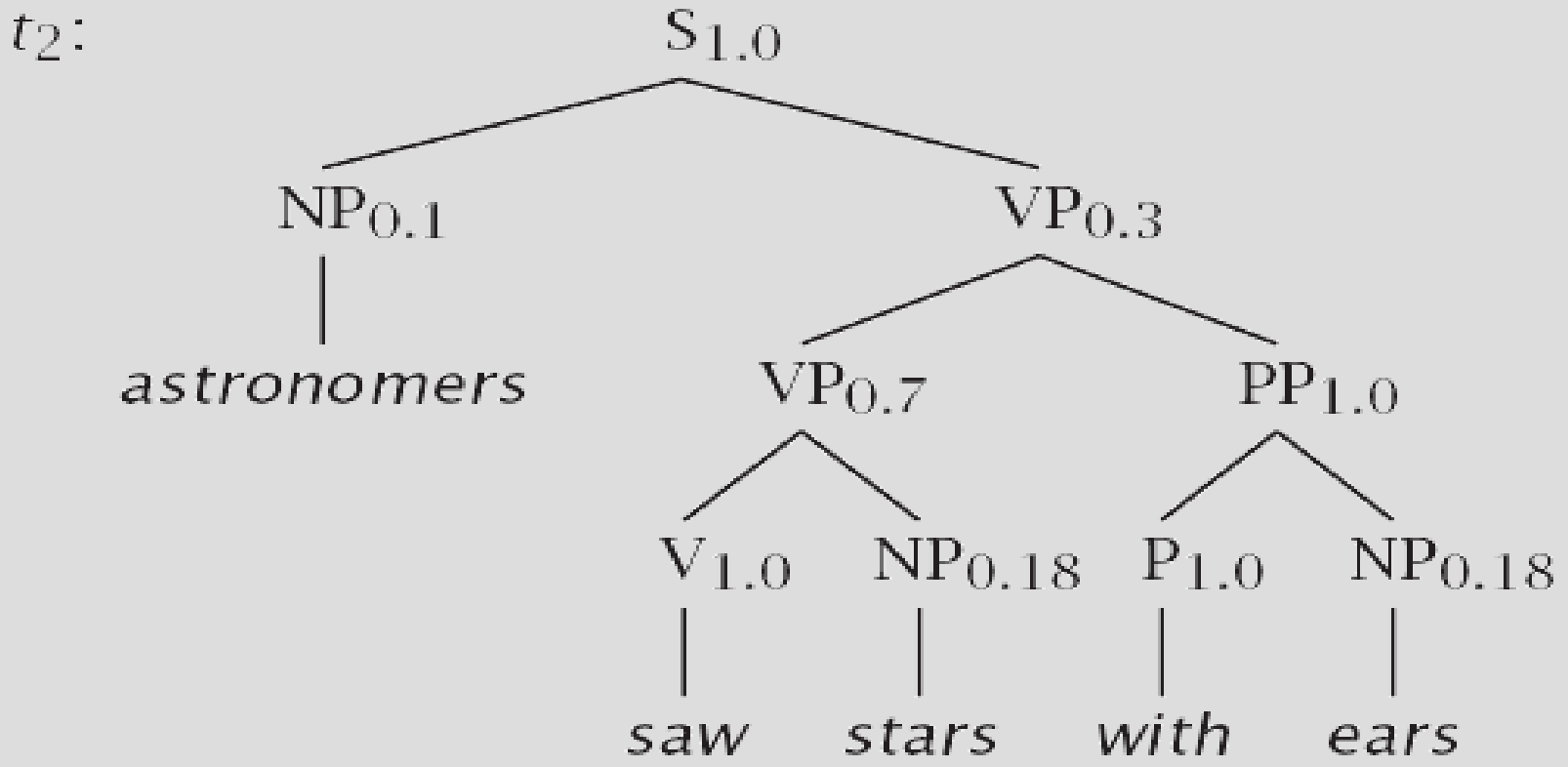
# Beispiel PCFG




$P(t_1)$ = 1.0 x 0.1 x 0.7 x 1.0 x 0.4 x 0.18 x 1.0 x 1.0 x 0.18

= 0.0009072

# Beispiel PCFG




$P(t_2)$ = 1.0 x 0.1 x 0.3 x 0.7 x 1.0 x 0.18 x 1.0 x 1.0 x 0.18

= 0.0006804

# Bilexikalisierte Grammatiken (Eisner 97 & Eisner u. Satta 99)

- Bisher Regeln der Form
  - $A \rightarrow A\ B$
  - $A \rightarrow x\ B$
  - $A \rightarrow x$
- Jetzt: Regeln mit zwei lexikalischen Köpfen
  - $A[x] \rightarrow B[x]\ C[y]$
  - $A[x] \rightarrow B[y]\ C[x]$
  - $A[x] \rightarrow x$

---

| | |
|---|---|
| A[x], B[y], C[y]..... | Nicht-Terminale |
| A, B, C.... | „traditionelle“ Nicht-Terminale |
| x, y, z.... | Wörter |

# Bilexikalisierte Grammatiken

- Beispiel
  - S[löst] → NP[Peggy] VP[löst]
  - VP[löst] → V[löst] NP[Puzzle]
  - NP[Puzzle] → Det[ein] N[Puzzle]
- Dadurch ausschliessen sinnloser Ableitungen
  - VP[löst] → V[löst] NP[Ziege]
  - NP[Puzzle] → Det[zwei] N[Puzzle]

# Parsen von bCFG mit CKY

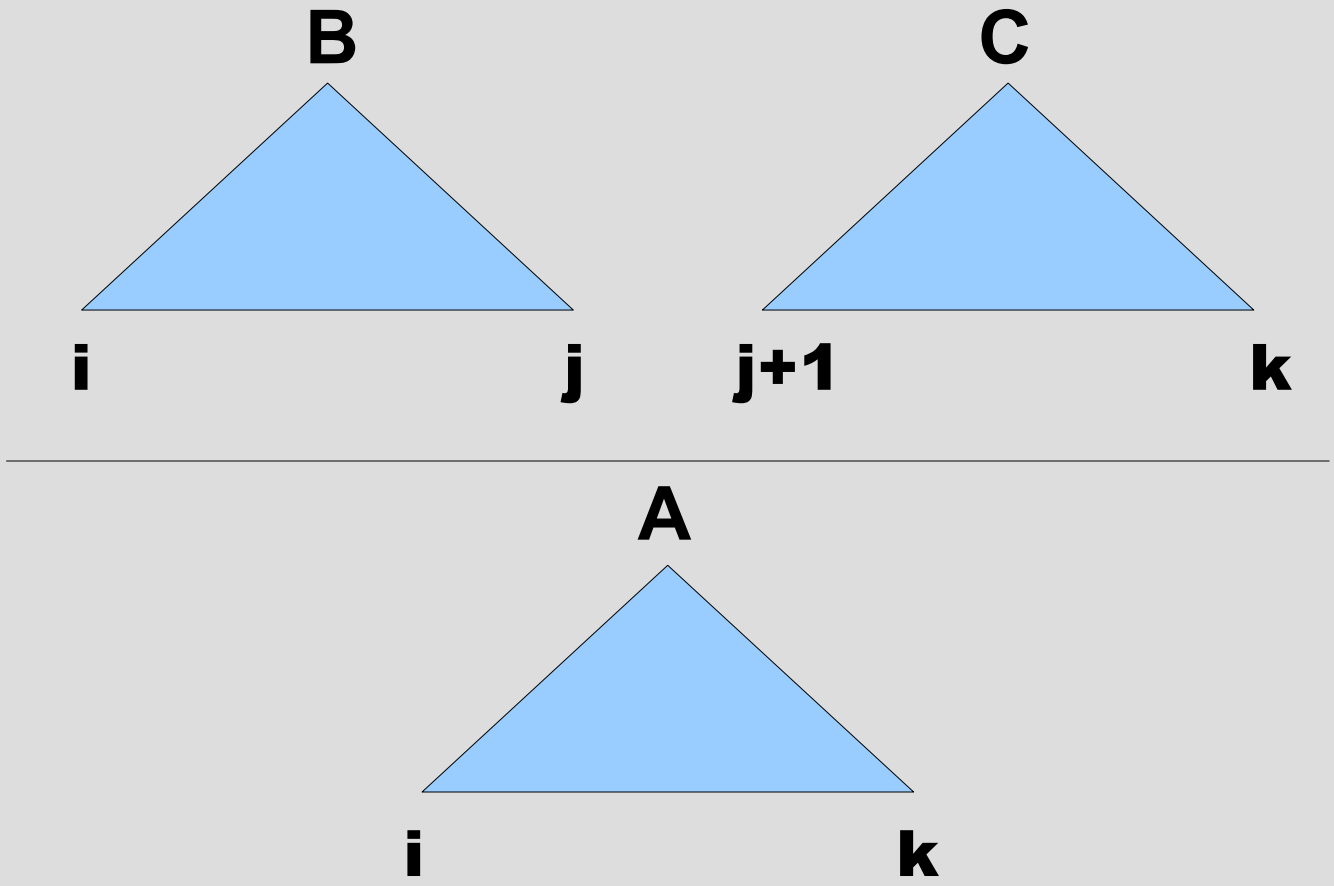
- Grösse der Grammatik = $O(t^3V^2)$
  mit $t = |\{A, B, C,...\}|$ $V = |\{x, y, z,...\}|$

  => Laufzeit CKY liegt in $O(t^3n^3V^2)$

- Für einen gegebene Eingabe gilt V = n

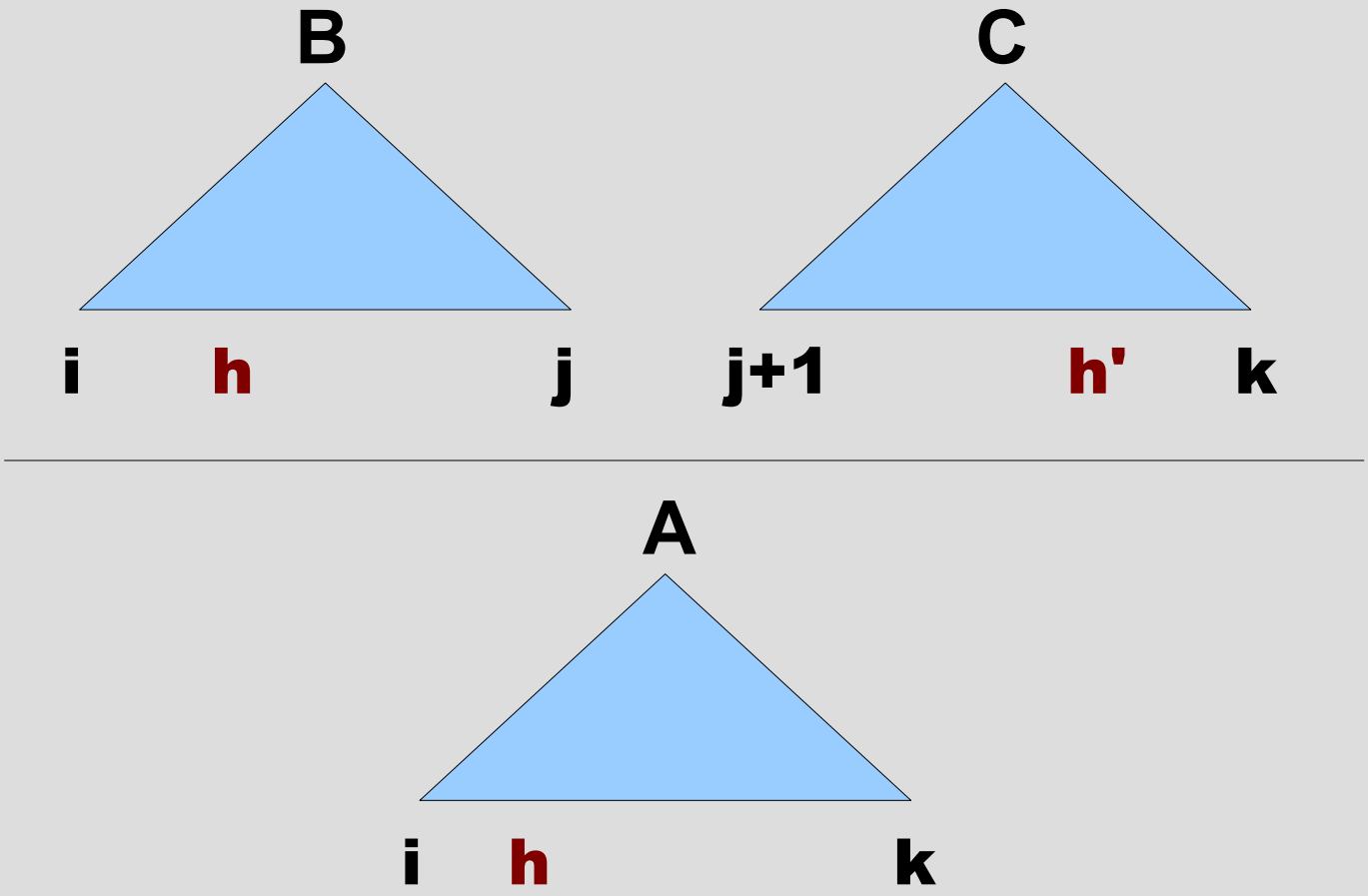
  => Laufzeit CKY liegt in $O(n^5t^3)$

# Warum ist CKY $O(n^5)$?




Für CFG: i, j und k variabel => $O(n^3$ Kombinationen)

# Warum ist CKY $O(n^5)$?

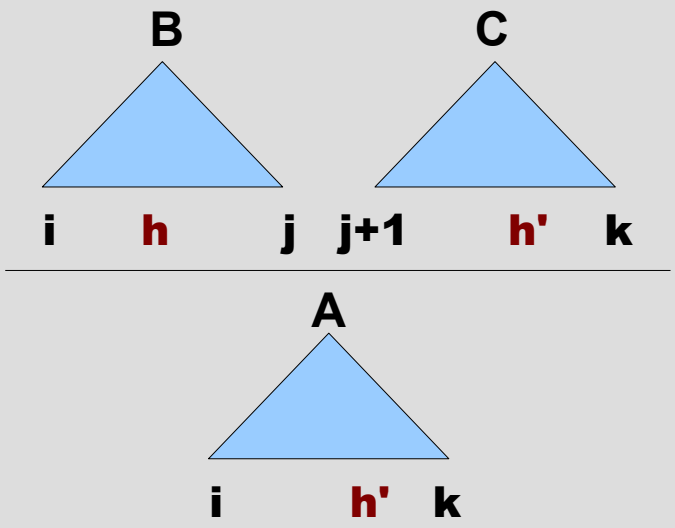



Für bCFG: i, j, k, h und h' variabel => $O(n^5$ Kombinationen)

# Idee 1:

**Welche C kommen für ein bestimmtes B in Frage?**

- Verschiedene Möglichkeiten für die Breite von C (k variiert)
- Verschiedene Möglichkeiten für den Kopf von C (h' variiert)

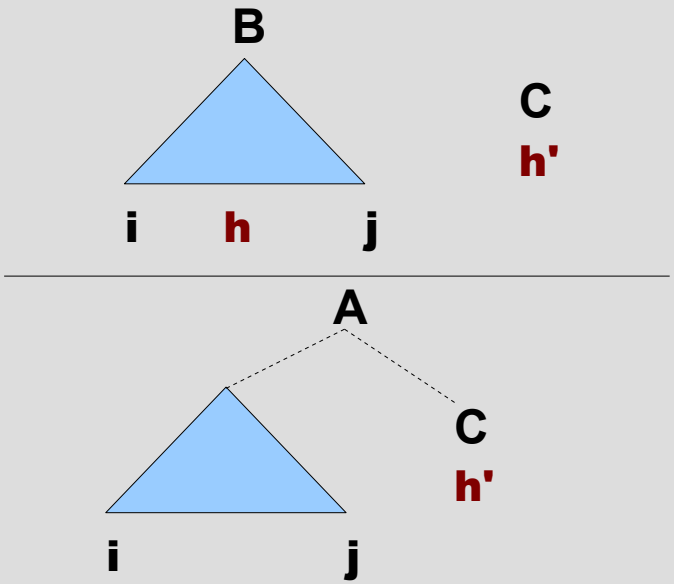



Mit CKY (nur ein Schritt) ergeben sich $n^2$ Möglichkeiten.

Versuche, die beiden Variations-Möglichkeiten in zwei Schritte aufzuteilen, um in linearer Zeit zu bleiben.

# Schritt 1 : h' variieren

- Kombiniere B nur mit dem Kopf von C

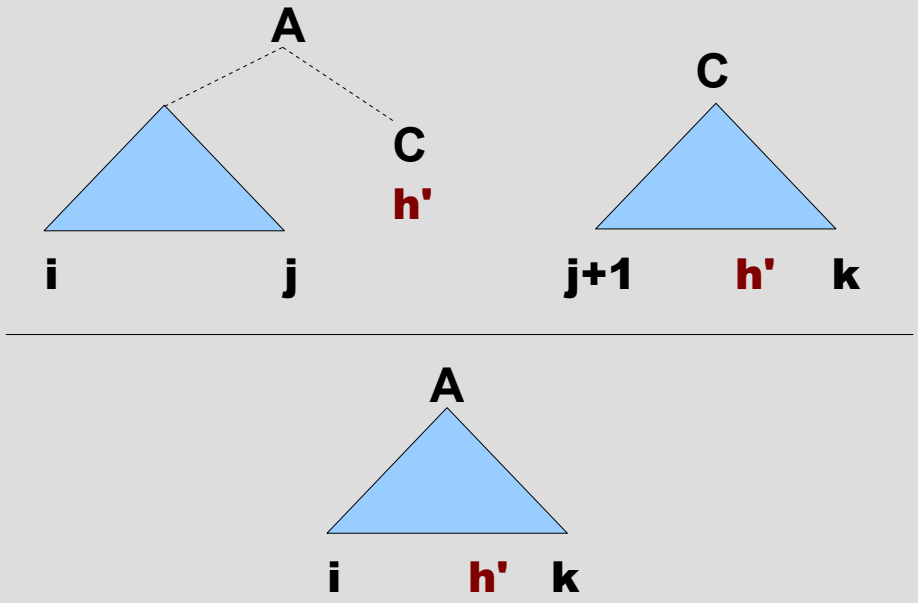



$A[d_{h'}] \rightarrow B[d_h]\ C[d_{h'}]$

=> nur 4 Variablen. Daher in $O(n^4)$ möglich.

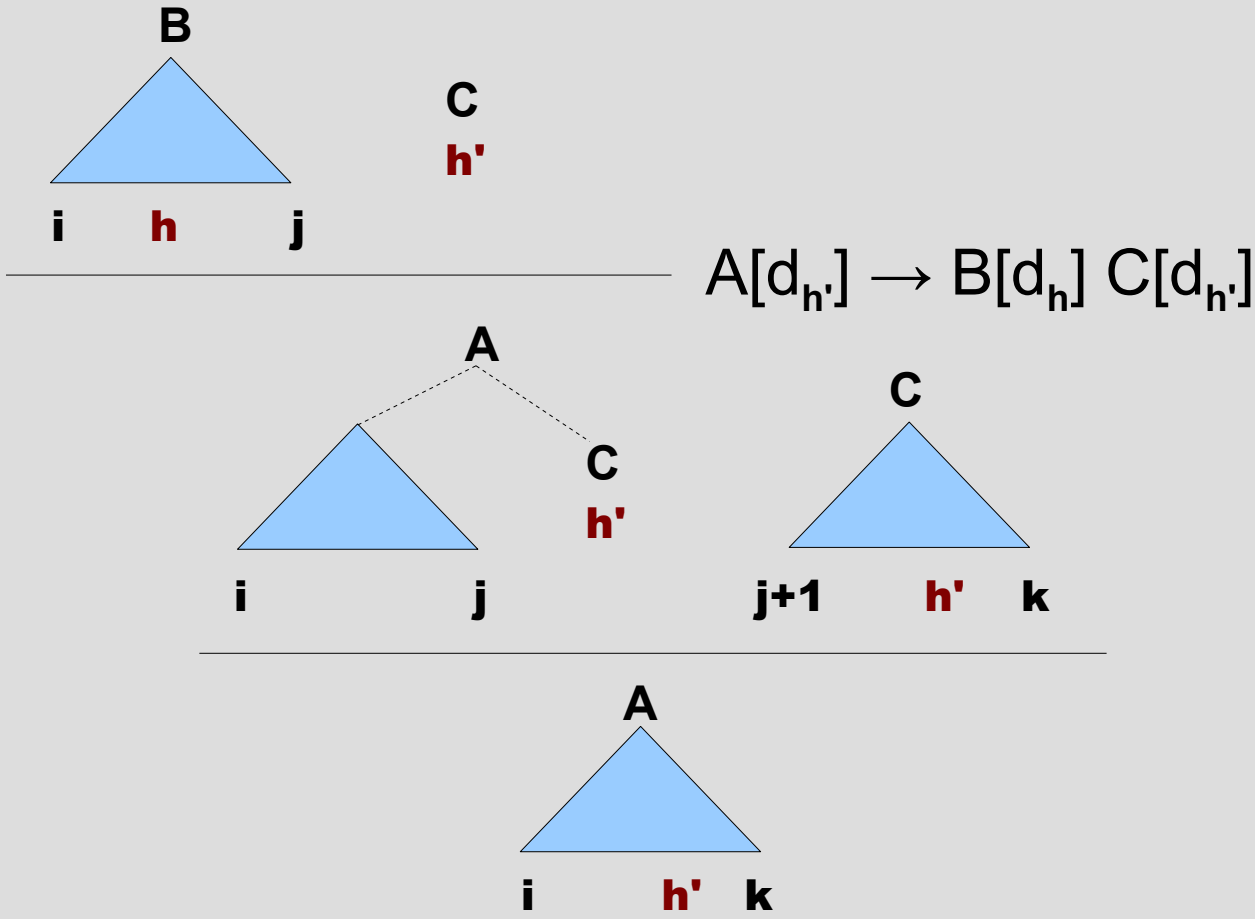
# Schritt 2 : k variieren

- Finde C's verschiedener Breite, die A\C vervollständigen




=> nur 4 Variablen. Daher in $O(n^4)$ möglich.

# Kombination der beiden Schritte




Beide Schritte in $O(n^4)$ => $O(2n^4) = O(n^4)$
Geht es noch schneller?

# Head Automaton Grammars (Alshawi 1996)

- HAG = Funktion, die jedem Terminal einen Head Automaton (HA) zuordnet.
- Ein einzelner HA akzeptiert eine Sprache von String-Paaren $\langle z_l, z_r \rangle \in V_T^* \times V_T^*$.
- Ein HA $H_a$ ist geteilt (split), wenn es keinen Zustand gibt, der mit einem ← - Übergang betreten und mit einem → - Übergang verlassen werden kann.
- Eine HAG H, bei der alle $H_a$ geteilt sind, entspricht einer bCFG, in der alle Ableitungen A[a] =>* xay die Form A[a] =>*xB[a] =>* xay haben.
- Ein solches H existiert für alle linguistisch relevanten Grammatiken.

# Beispiel HAG

[Good old Peggy] **solved** [the puzzle] [with her teeth]

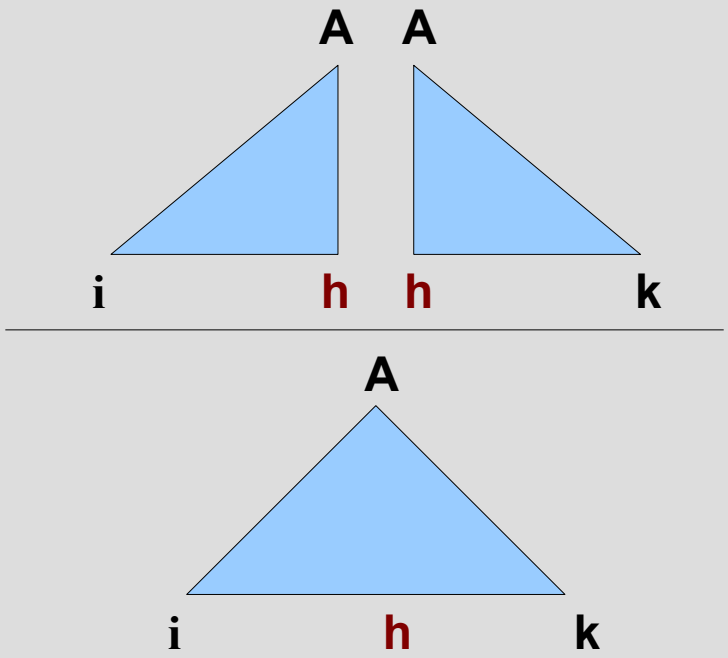
Der HA für **solved**:

- Kann adjazente Wörter auf beiden Seiten konsumieren.
- Aber erst, nachdem diese ihre Dependenten konsumiert haben.

| | |
|---|---|
| [Peggy]**solved**[puzzle][with] | (V) |
| [Peggy]**solved**[puzzle] | (VP) |
| [Peggy]**solved** | (VP) |
| **solved** | (S) |

# Split Grammars

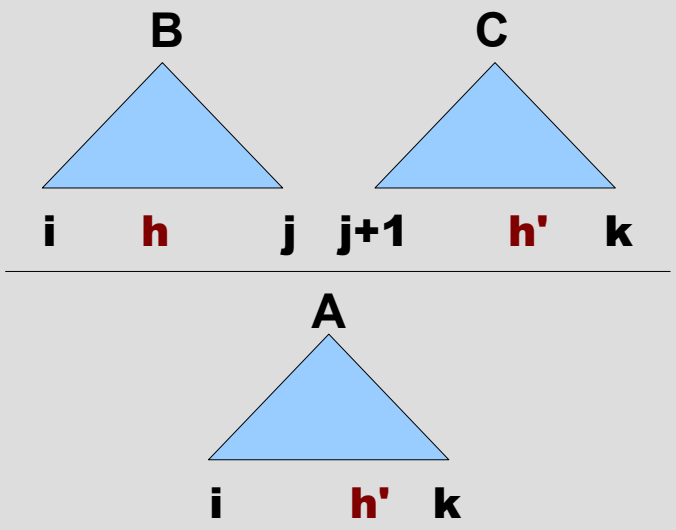
- Eine Split Grammar erlaubt folgende Regel:



# Idee 2:

Welche B und C kann man kombinieren?

- Verschiedene Möglichkeiten für die Breite von C (k variiert)
- Verschiedene Möglichkeiten für den Mittelpunkt (j variiert)

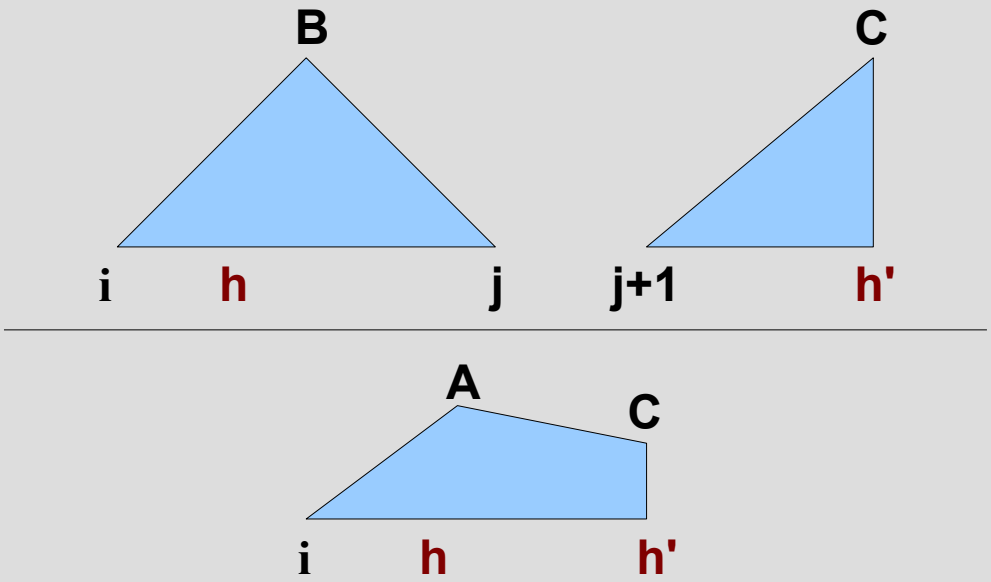



Mit CKY (nur ein Schritt) ergeben sich $n^2$ Möglichkeiten.

Versuche wieder, die beiden Variationsmöglichkeiten in zwei Schritte aufzuteilen, um in linearer Zeit zu bleiben.

# Schritt 1: j variieren

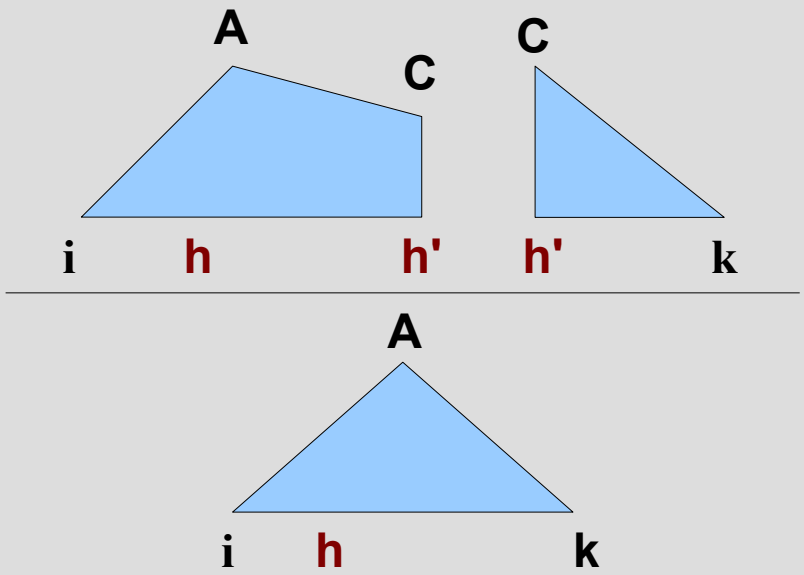
- Kombiniere B nur mit der linken Hälfte von C




=> nur 4 Variablen. Daher in $O(n^4)$ möglich.

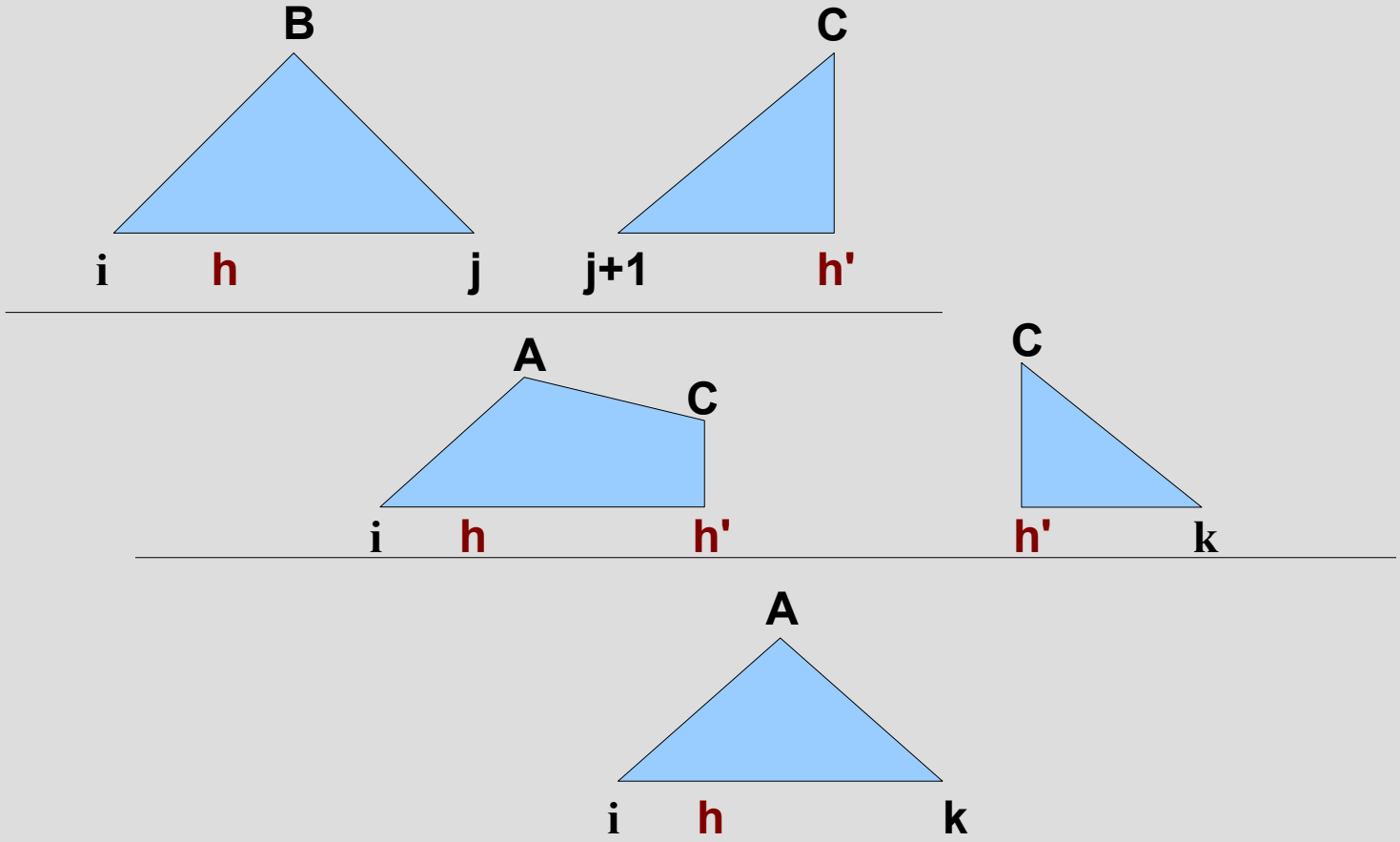
# Schritt 2: variiere k

- Hänge die rechte Hälfte von C an




=> nur 4 Variablen. Daher in $O(n^4)$ möglich.

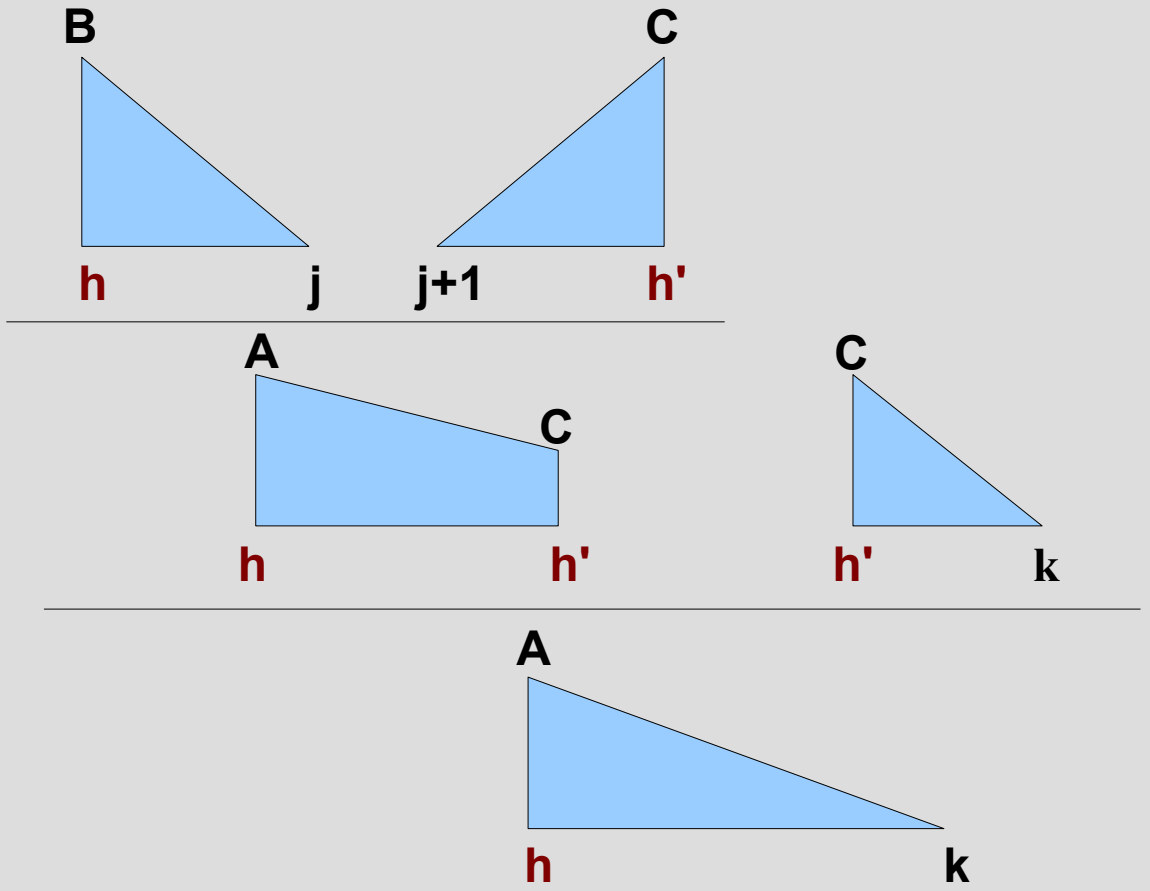
# Kombination der beiden Schritte:




- Beide Schritte in $O(n^4)$ => $O(2n^4) = O(n^4)$

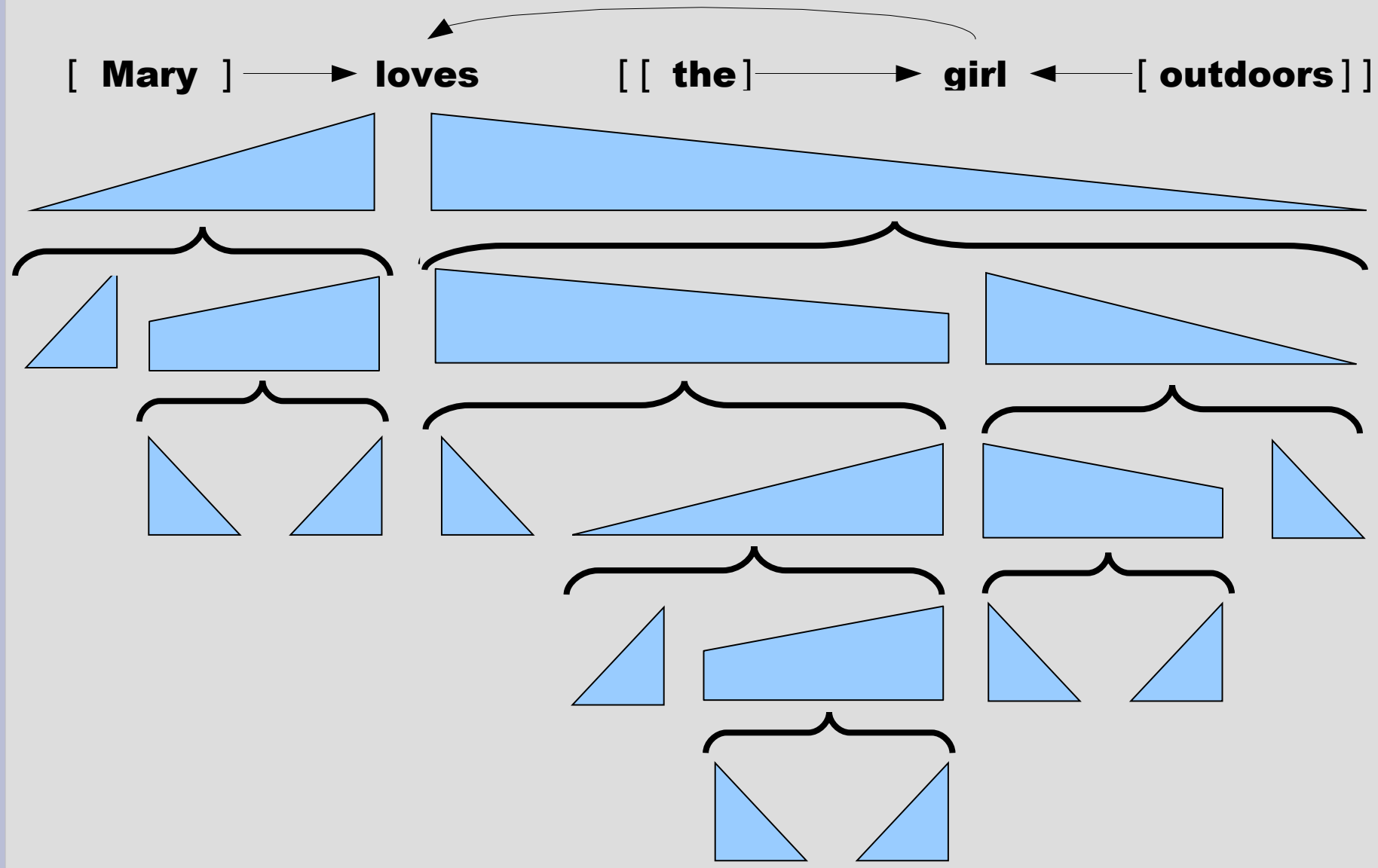
# Verbesserung von Idee 2

- Warte mit dem bestimmen von i




- Nur noch 3 Variablen => $O(n^3)$

# Beispiel für den O(n³) Halbbaum Algorithmus



# Theoretische Verbesserung der Laufzeit

- n = Eingabelänge
  t = traditionelle Nichtterminale oder Zustände

- Naiv: $O(n^5 g^2 t)$
- Idee 1: $O(n^4 g^2 t)$
- Für split grammars:
  - Eisner (1997): $O(n^3 t^4)$
  - Idee 2: $O(n^3 t^3)$

Beachte: Alle Laufzeiten sind unabhängig von der Grösse des Vokabulars!

# Verbesserung der Laufzeit in der Praxis

- Parsen mit pruning (n = 30):
  - Beide $O(n^3)$-Algorithmen 5x schneller als der naive $O(n^5)$-Algorithmus.
- Parsen ohne pruning (n = 30):
  - Der alte $O(n^3)$-Algorithmus läuft 3x schneller.
  - Der neue $O(n^3)$-Algorithmus sogar 19x schneller.

# Zusammenfassung

- Einführung in bilexikalisierte Grammatiken
- Verbesserung des naiven Parsing-Algorithmus für (allgemeine) bCFG. Jetzt Laufzeit in $O(n^4)$ statt $O(n^5)$
- Für Spezialfälle (split grammars) sogar in $O(n^3)$
- Der theoretische Geschwindigkeitszuwachs wurde in der Praxis bestätigt.
- Für bilexikalisierte TAGs: Verbesserung des naiven $O(n^8)$-Algorithmus auf $O(n^7)$.

# Literatur

- *„Efficient Parsing for Bilexical CFGs and HAG“*
  *Jason Eisner & Giorgio Satta (1999)*
- *„Bilexical Grammars and their cubic-time Parsing Algorithms“*
  Jason Eisner (2000)
- *„Head automata and bilingual tiling. Translations with minimal representations.“*
  H. Alshawi (1996)