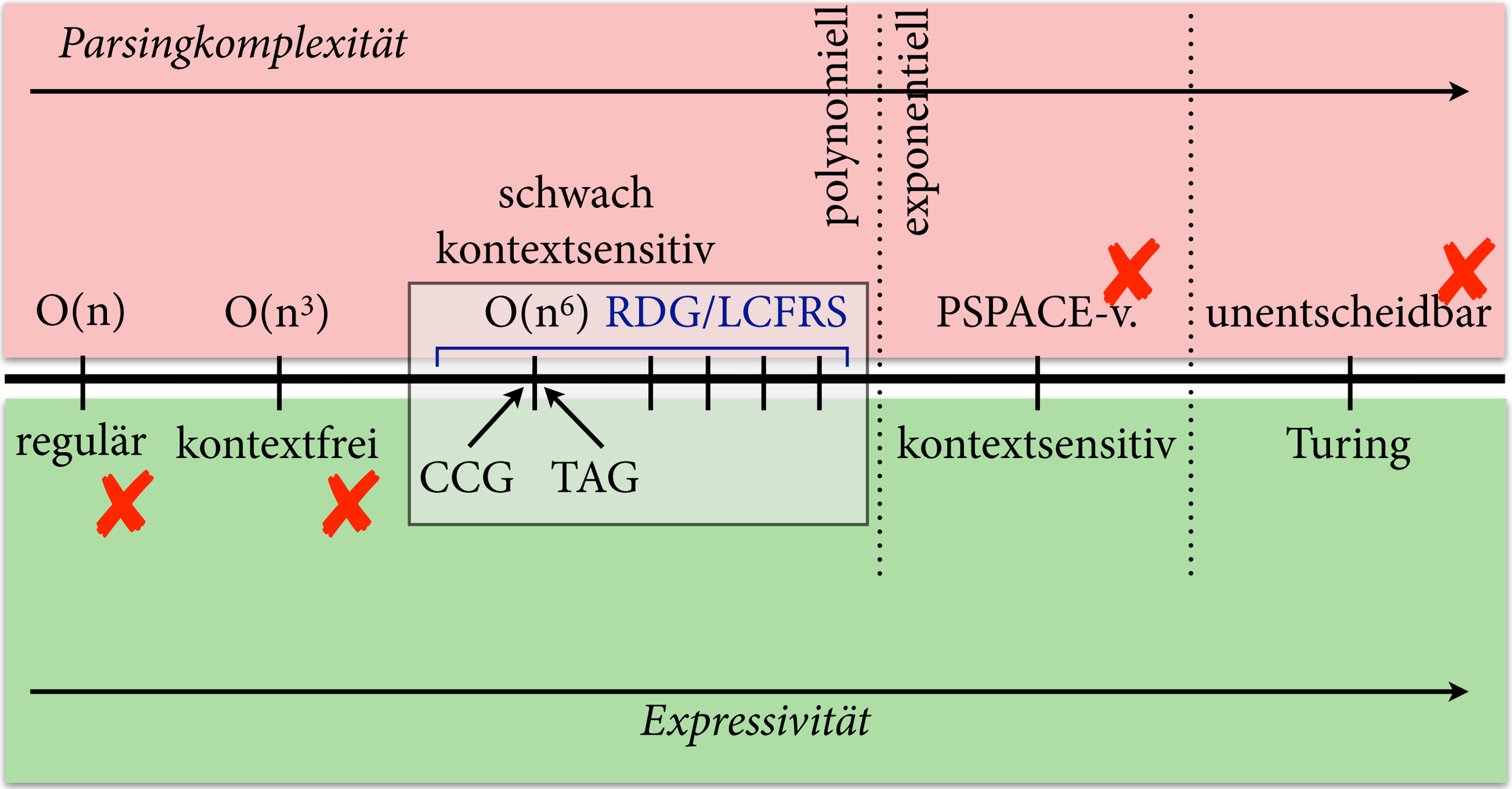
(Interpretierte)
Reguläre Baumgrammatiken

Vorlesung “Grammatikformalismen”

Alexander Koller

2. Juni 2017

Grammatikformalismen



Bäume

- Statt Alphabeten haben wir jetzt *Signaturen:*
  - endliche Menge von Zeichen, die man als Knotenlabels in Bäumen verwenden kann
  - jedes Zeichen f hat eine *Arität/Stelligkeit* n

- *Baum* über einer Signatur Σ:
  - jeder Knoten hat ein Label $f \in \Sigma$
  - wenn Label von Knoten *u* Arität *n* hat, dann hat *u* genau *n* Kinder im Baum

- T_Σ = alle Bäume über Σ

Reguläre Baumgrammatiken

- Reguläre Baumgrammatik (RTG) ist ein Tupel $G = (\Sigma, N, S, P)$, wobei
  - Σ eine Signatur (= Terminalsymbole)
  - N eine endliche Menge von Nichtterminalsymbolen
  - $S \in N$ das Startsymbol
  - P eine Menge von Produktionsregeln von der Form $A \to f(A_1, ..., A_n)$, wobei $f \in \Sigma$ und $A, A_1, ..., A_n \in N$.

- Eine RTG G definiert eine *Baumsprache* $L(G) \subseteq T_\Sigma$.

Ableitungen von RTGs

- Ableitungsprozess:
  - mit Startsymbol anfangen
  - in jedem Schritt ein Nichtterminalsymbol durch Baum auf der rechten Seite einer Regel ersetzen
  - wenn der Baum nur noch Terminalsymbole enthält, kommt er in die Sprache.

$S \to f(A,S)$
$S \to c$
$A \to a$
$A \to b$

RTG G

Ableitungen von RTGs

- Ableitungsprozess:
  - mit Startsymbol anfangen
  - in jedem Schritt ein Nichtterminalsymbol durch Baum auf der rechten Seite einer Regel ersetzen
  - wenn der Baum nur noch Terminalsymbole enthält, kommt er in die Sprache.

$S \to f(A,S)$
$S \to c$
$A \to a$
$A \to b$

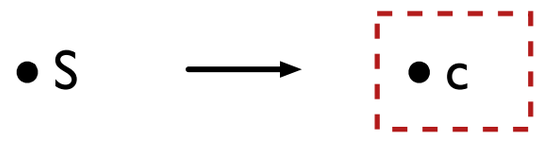
RTG G

Ableitungen von RTGs

- Ableitungsprozess:
  - mit Startsymbol anfangen
  - in jedem Schritt ein Nichtterminalsymbol durch Baum auf der rechten Seite einer Regel ersetzen
  - wenn der Baum nur noch Terminalsymbole enthält, kommt er in die Sprache.

$S \to f(A,S)$
$S \to c$
$A \to a$
$A \to b$

RTG G

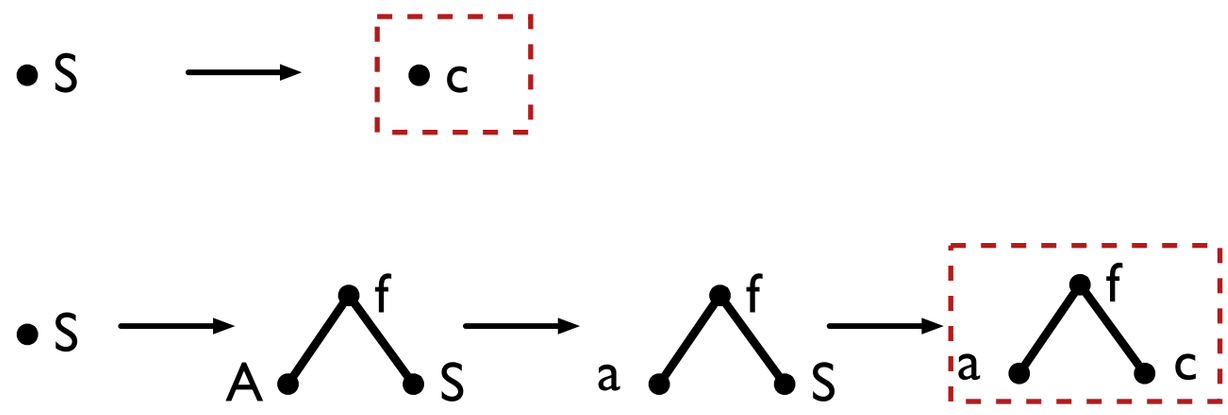


Ableitungen von RTGs

- Ableitungsprozess:
  - mit Startsymbol anfangen
  - in jedem Schritt ein Nichtterminalsymbol durch Baum auf der rechten Seite einer Regel ersetzen
  - wenn der Baum nur noch Terminalsymbole enthält, kommt er in die Sprache.

$S \to f(A,S)$
$S \to c$
$A \to a$
$A \to b$

RTG G

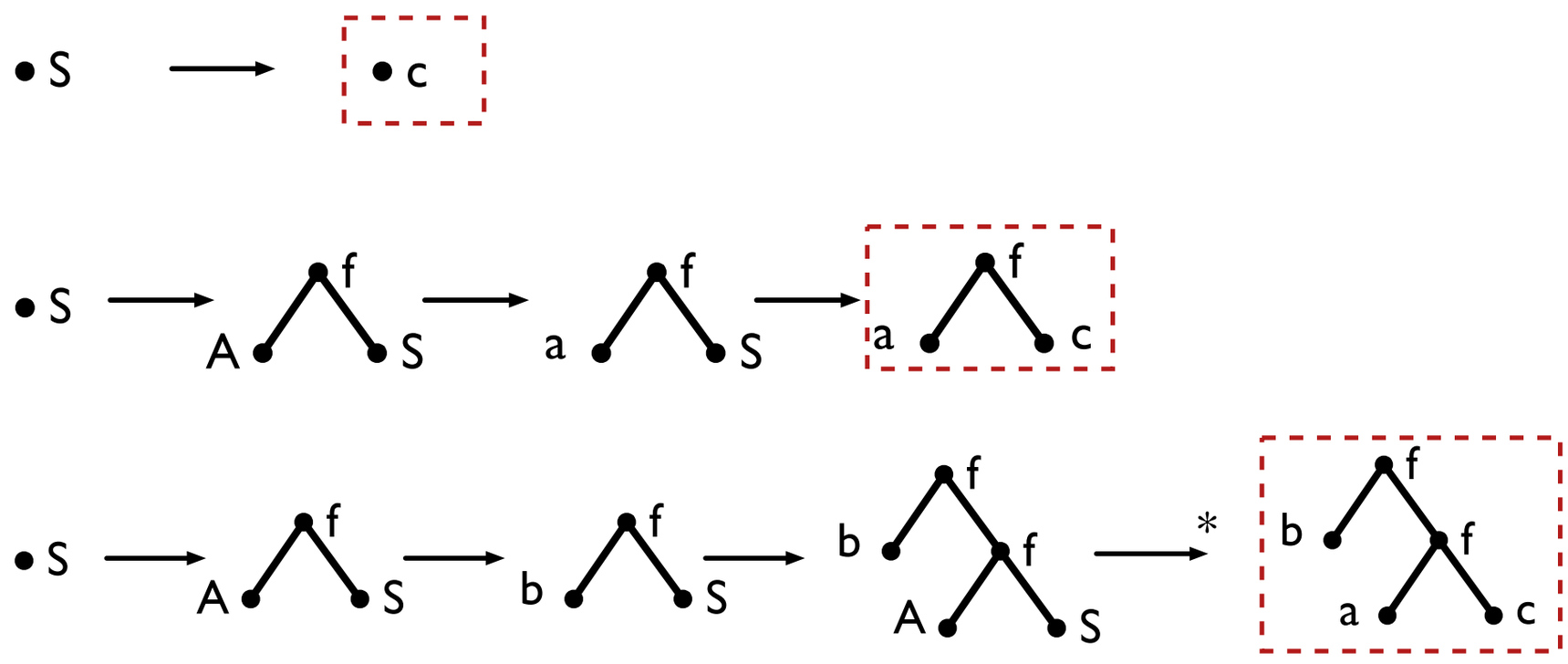


Ableitungen von RTGs

- Ableitungsprozess:
  - mit Startsymbol anfangen
  - in jedem Schritt ein Nichtterminalsymbol durch Baum auf der rechten Seite einer Regel ersetzen
  - wenn der Baum nur noch Terminalsymbole enthält, kommt er in die Sprache.

$S \to f(A,S)$
$S \to c$
$A \to a$
$A \to b$

RTG G



kfG ⇔ RTG

- Für jede kfG G gilt: Die Sprache T(G) der Parsebäume von G ist eine reguläre Baumsprache.

kfG	RTG
S → NP VP	S → s_2(NP,VP)
NP → Hans	NP → np_1($hans_0$)
VP → schläft	VP → vp_1($schläft_0$)

- Für jede RTG G gilt: Die Sprache der String-Erträge der Bäume in L(G) ist eine kontextfreie Stringsprache.

Abschlusseigenschaften

- Reguläre Baumsprachen sind abgeschlossen unter
  - Schnitt
  - Vereinigung
  - Komplement
  - Vorwärts-Anwendung von linearen Baumhomomorphismen
  - Rückwärts-Anwendung von beliebigen Baumhomomorphismen

Abschluss unter Schnitt

mit Produktkonstruktion (vgl. endliche Stringautomaten)

$S \rightarrow s_2(NP, VP)$

$VP \rightarrow vp_2(V, NP)$

$NP \rightarrow hans_0$

$V \rightarrow isst_0$

$NP \rightarrow kekse_0$

G_1

$[0,3] \rightarrow s_2([0,1], [1,3])$

$[0,3] \rightarrow s_2([0,2], [2,3])$

$[1,3] \rightarrow vp_2([1,2], [2,3])$

$[0,1] \rightarrow hans_0$

$[1,2] \rightarrow isst_0$

$[2,3] \rightarrow kekse_0$

G_2

$S[0,3] \rightarrow s_2(NP[0,1], VP[1,3])$

$VP[1,3] \rightarrow vp_2(V[1,2], NP[2,3])$

$NP[0,1] \rightarrow hans_0$

$V[1,2] \rightarrow isst_0$

$NP[2,3] \rightarrow kekse_0$

G_3

mit $L(G_3) = L(G_1) \cap L(G_2)$

Baumautomaten

- Ein (bottom-up) endlicher Baumautomat ist ein Tupel $A = (\Sigma, Q, q_F, R)$, wobei
  - Σ eine Signatur (= Terminalsymbole)
  - Q eine endliche Menge von Zuständen
  - $q_F \in Q$ der Endzustand
  - R eine Menge von Übergangsregeln von der Form $f(q_1, ..., q_n) \to q$, wobei $f \in \Sigma$ und $q, q_1, ..., q_n \in Q$.

- Ein Automat *akzeptiert* einen Baum t, wenn er an der Wurzel in Zustand q_F sein kann.
  Sprache $L(A)$ = Menge der akzeptierten Bäume.

Baumautomaten

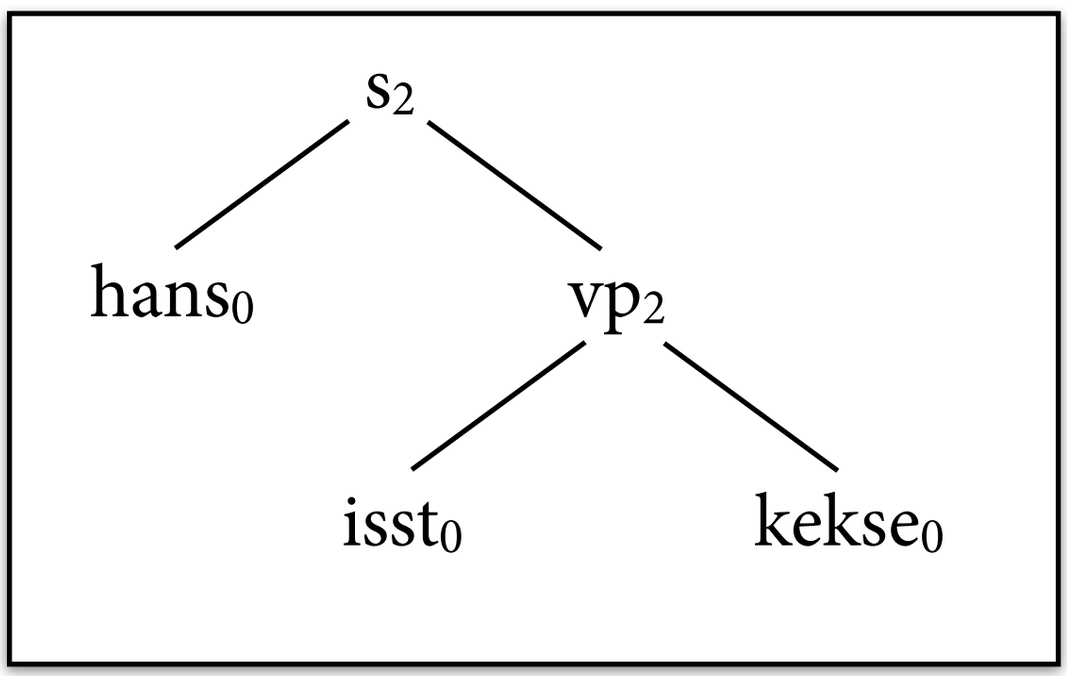
$s_2(q_{NP}, q_{VP}) \rightarrow q_S$

$vp_2(q_V, q_{NP}) \rightarrow q_{VP}$

$hans_0 \rightarrow q_{NP}$

$isst_0 \rightarrow q_V$

$kekse_0 \rightarrow q_{NP}$



A t

Baumautomaten

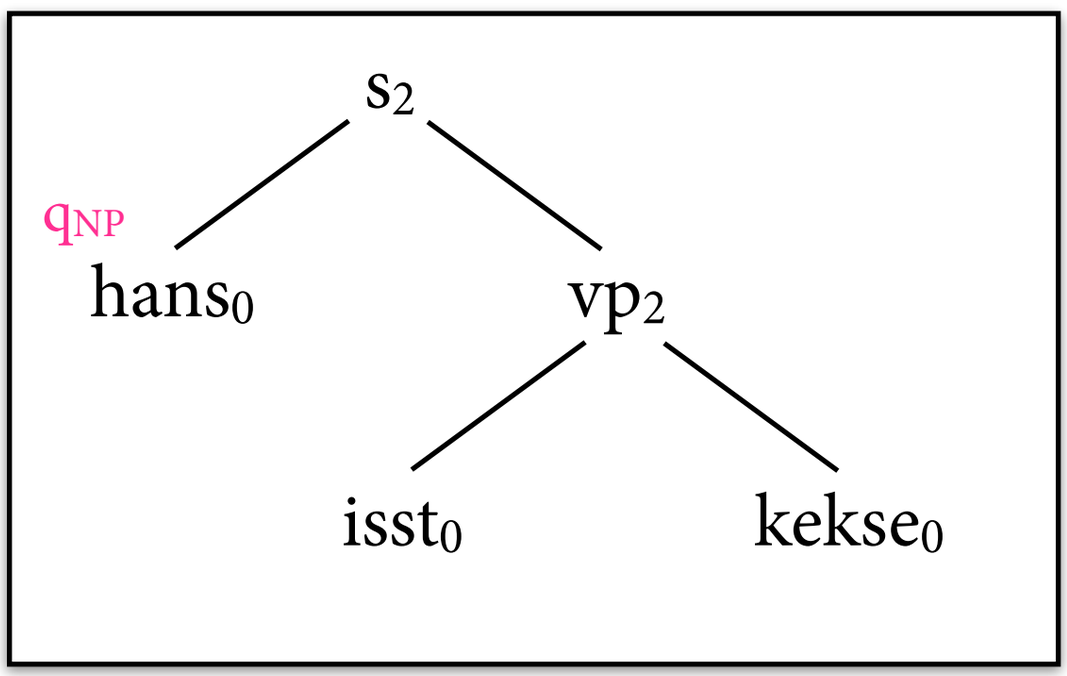
$s_2(q_{NP}, q_{VP}) \rightarrow q_S$

$vp_2(q_V, q_{NP}) \rightarrow q_{VP}$

$hans_0 \rightarrow q_{NP}$

$isst_0 \rightarrow q_V$

$kekse_0 \rightarrow q_{NP}$



A t

Baumautomaten

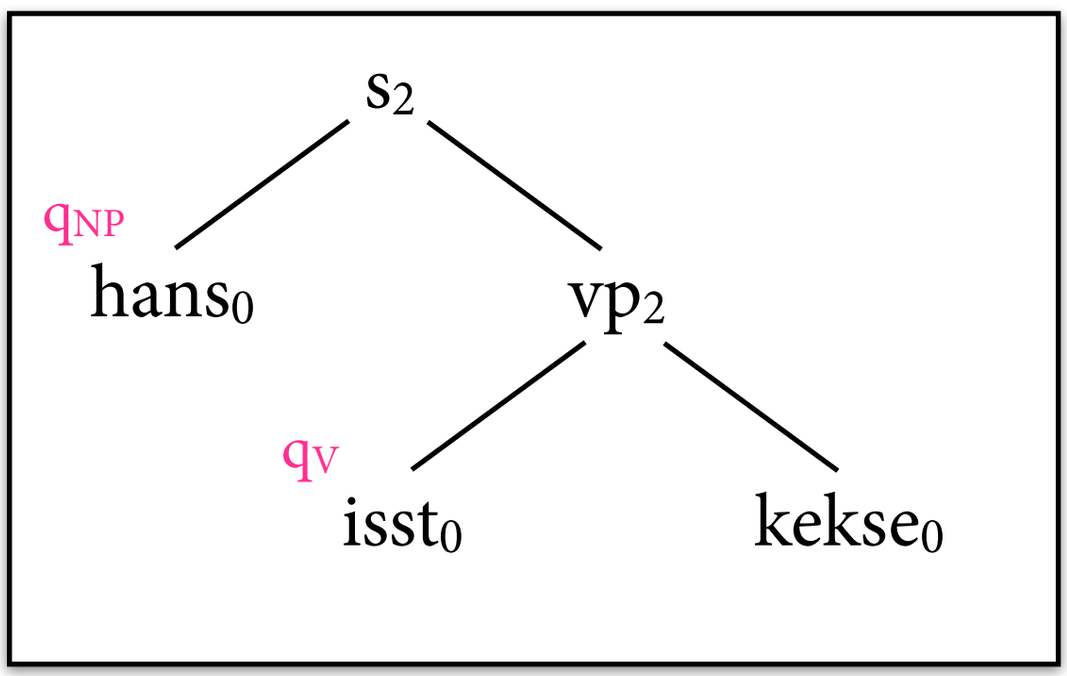
$s_2(q_{NP}, q_{VP}) \rightarrow q_S$

$vp_2(q_V, q_{NP}) \rightarrow q_{VP}$

$hans_0 \rightarrow q_{NP}$

$isst_0 \rightarrow q_V$

$kekse_0 \rightarrow q_{NP}$



A t

Baumautomaten

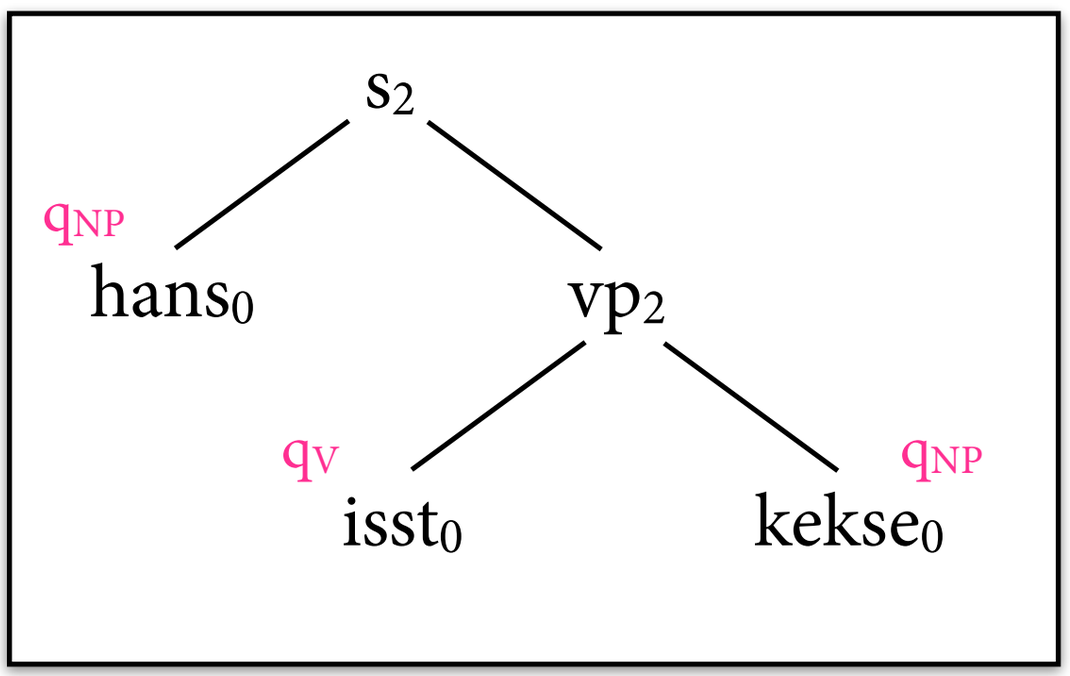
$s_2(q_{NP}, q_{VP}) \rightarrow q_S$

$vp_2(q_V, q_{NP}) \rightarrow q_{VP}$

$hans_0 \rightarrow q_{NP}$

$isst_0 \rightarrow q_V$

$kekse_0 \rightarrow q_{NP}$



A t

Baumautomaten

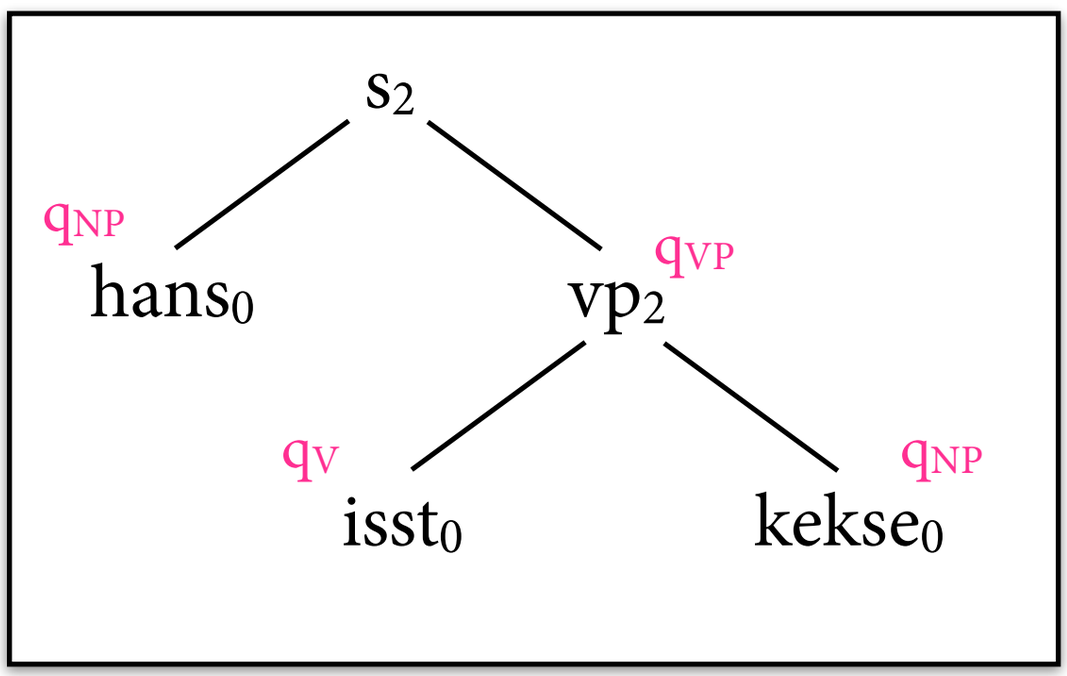
$s_2(q_{NP}, q_{VP}) \rightarrow q_S$

$vp_2(q_V, q_{NP}) \rightarrow q_{VP}$

$hans_0 \rightarrow q_{NP}$

$isst_0 \rightarrow q_V$

$kekse_0 \rightarrow q_{NP}$



A t

Baumautomaten

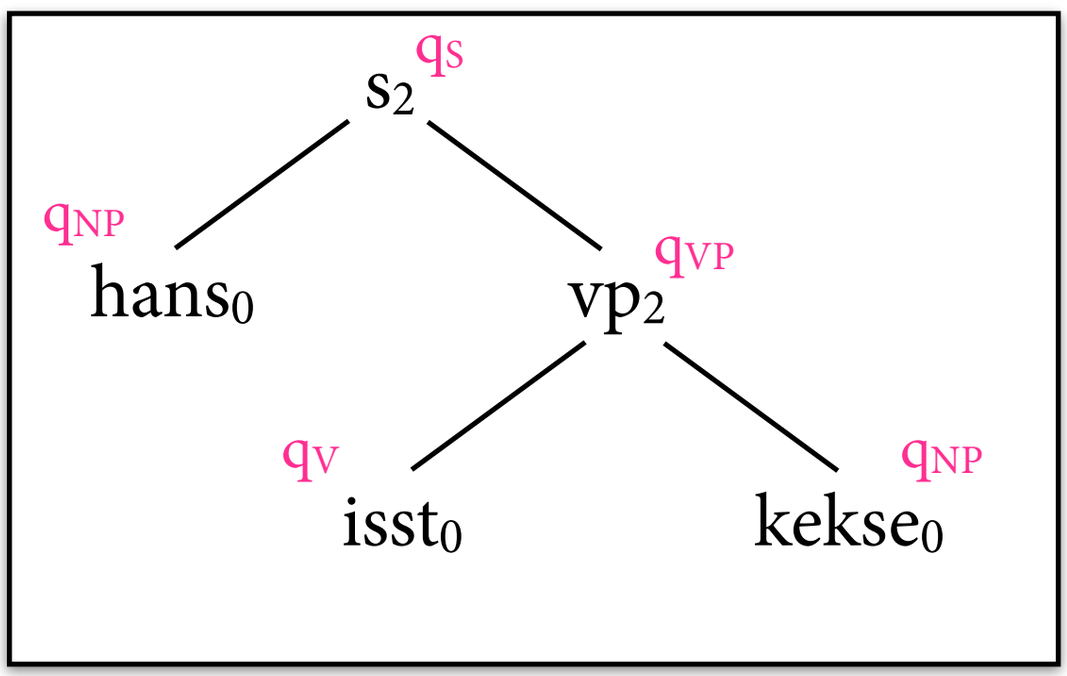
$s_2(q_{NP}, q_{VP}) \rightarrow q_S$

$vp_2(q_V, q_{NP}) \rightarrow q_{VP}$

$hans_0 \rightarrow q_{NP}$

$isst_0 \rightarrow q_V$

$kekse_0 \rightarrow q_{NP}$



A t

Baumautomaten

$s_2(q_{NP}, q_{VP}) \rightarrow q_S$

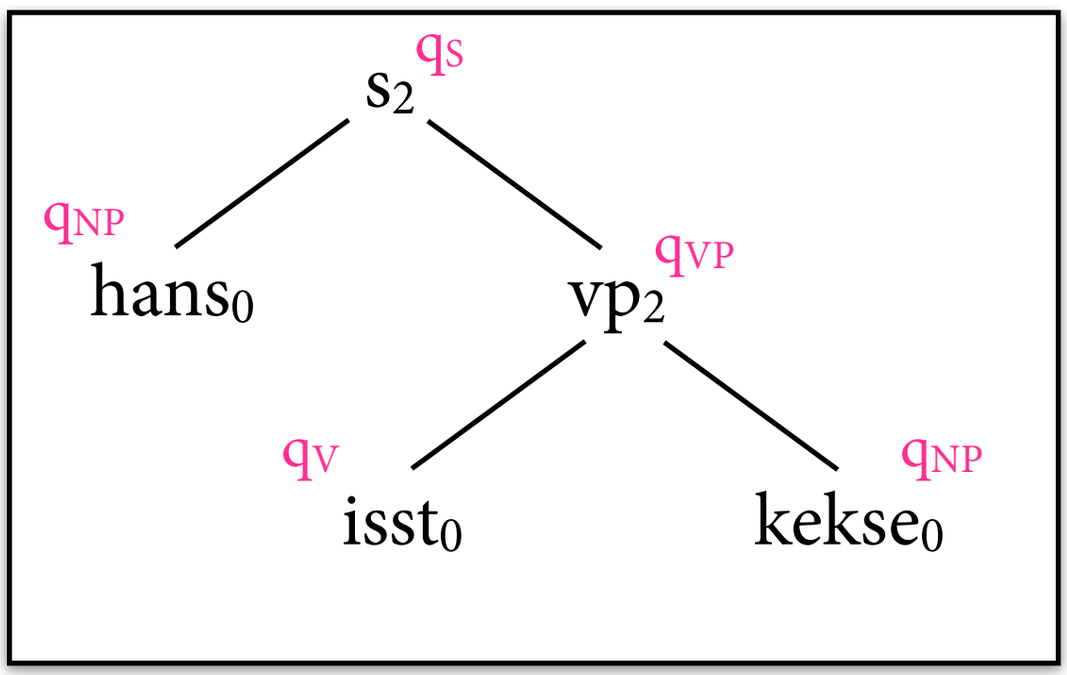
$vp_2(q_V, q_{NP}) \rightarrow q_{VP}$

$hans_0 \rightarrow q_{NP}$

$isst_0 \rightarrow q_V$

$kekse_0 \rightarrow q_{NP}$

A



t

$\Rightarrow$ A akzeptiert t (falls $q_F = q_S$)

Baumautomaten

$s_2(q_{NP}, q_{VP}) \rightarrow q_S$

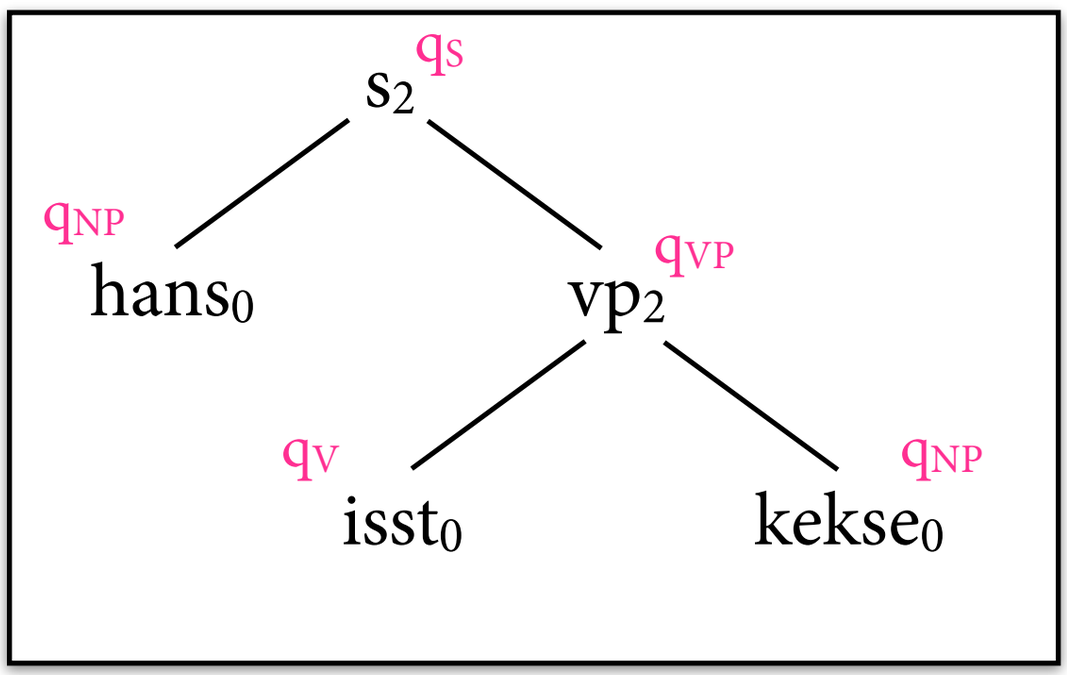
$vp_2(q_V, q_{NP}) \rightarrow q_{VP}$

$hans_0 \rightarrow q_{NP}$

$isst_0 \rightarrow q_V$

$kekse_0 \rightarrow q_{NP}$

A



t

$\Rightarrow$ A akzeptiert t (falls $q_F = q_S$)

$\Rightarrow t \in L(A)$

Automaten und Grammatiken

- RTGs und bottom-up-Automaten sind stark äquivalent, d.h. bu-Automaten akzeptieren genau die regulären Baumsprachen.

- Man kann *top-down*-Baumautomaten definieren; Regeln von bu-Automaten umdrehen, äquivalent.

- Caveat:
  - für jede RTL existiert *deterministischer* bu-Automat (max. 1 Regel pro Terminalsymbol + Kinderzustände)
  - aber nicht unbedingt ein deterministischer td-Automat

IRTGs

- *Interpretierte* reguläre Baumgrammatiken (IRTGs): Koller & Kuhlmann (2011).

- Kann sehr große Klasse von Grammatikformalismen stark äquivalent in IRTGs codieren.

- Verwenden Algorithmen auf Baumautomaten als Grundlage von sehr allgemeinen Parsingalgorithmen.

Algebra

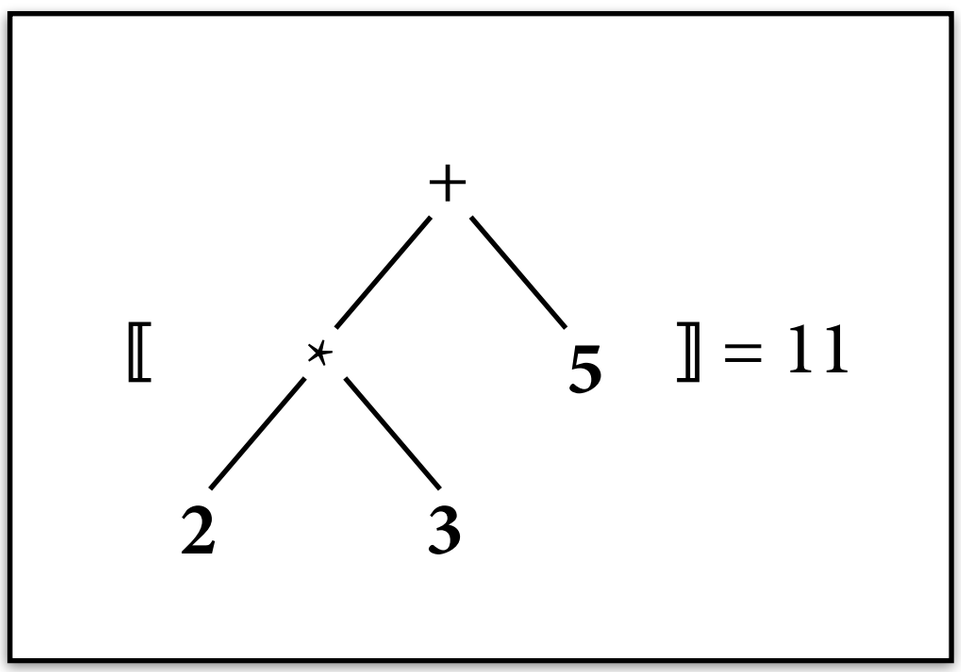
- Eine *Algebra* über der Signatur Σ ist eine Struktur $\mathbf{A} = (A, I)$, bestehend aus:
  - einer nichtleeren *Domäne* A (= Objekte der Algebra)
  - einer *Interpretationsfunktion* I, die jedem Symbol $f \in \Sigma$ mit Arität k eine Funktion $I(f): A^k \to A$ zuweist.
- Kann *Term* $t \in T_\Sigma$ als Wert von A *evaluieren:* $[\![f(t_1,\ldots,t_n)]\!] = I(f)\ ([\![t_1]\!], \ldots, [\![t_n]\!])$.

Beispiel: Arithmetik

- $\Sigma = \{+|_2, *|_2, \mathbf{1}|_0, \mathbf{2}|_0, \mathbf{3}|_0, \ldots\}$

- $\mathbf{N} = (\{1, 2, 3, \ldots\}, I_N)$ mit z.B.

  $I_N(+)(x,y) = x+y$
  $I_N(\mathbf{1}) = 1$ usw.

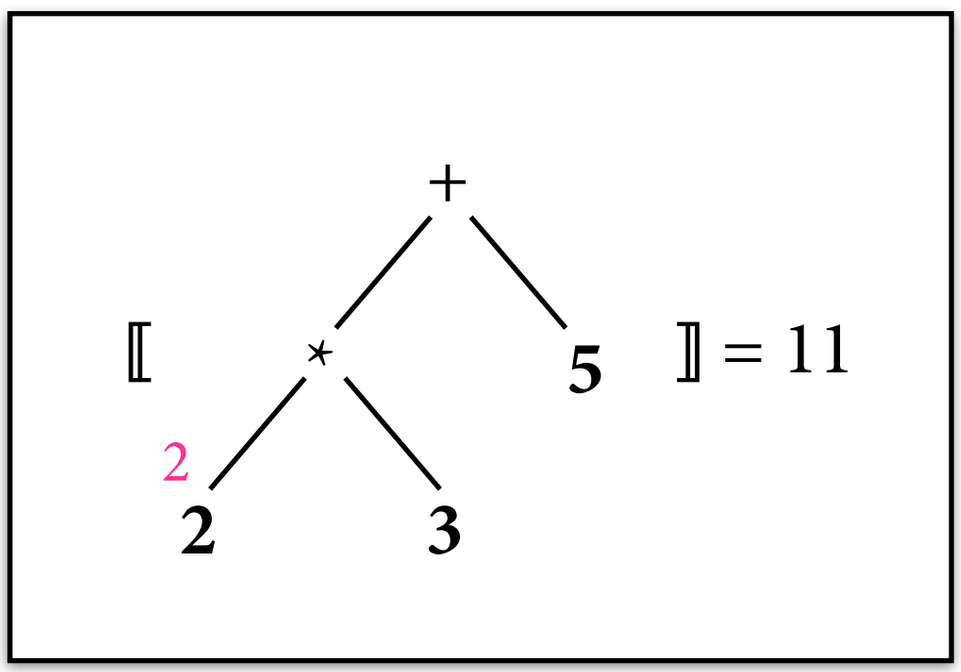


Beispiel: Arithmetik

- $\Sigma = \{+|_2, *|_2, \mathbf{1}|_0, \mathbf{2}|_0, \mathbf{3}|_0, \ldots\}$

- $\mathbf{N} = (\{1, 2, 3, \ldots\}, I_N)$ mit z.B.

  $I_N(+)(x,y) = x+y$
  $I_N(\mathbf{1}) = 1$ usw.

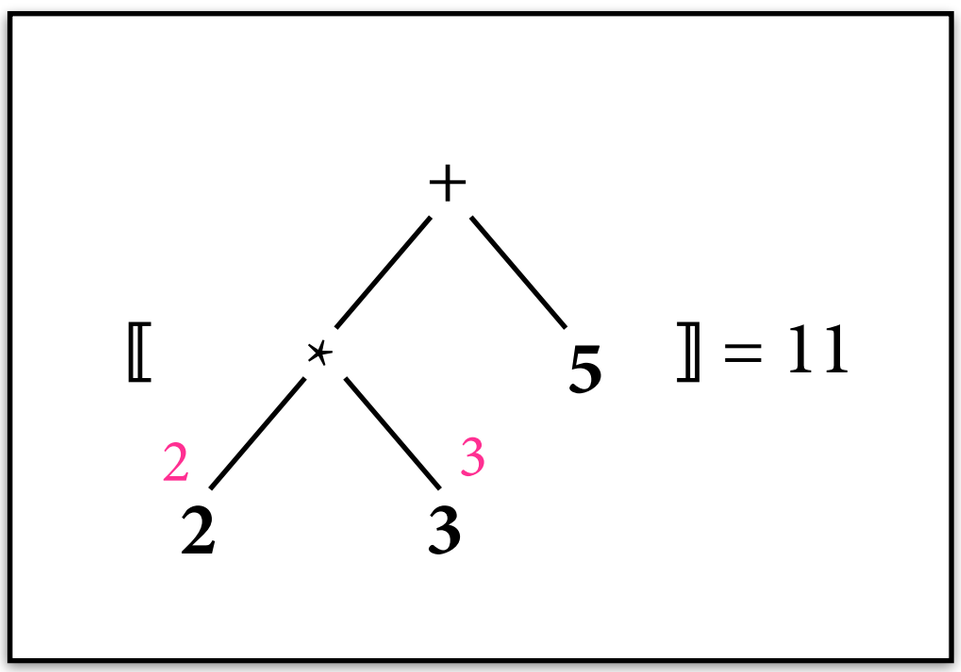


Beispiel: Arithmetik

- $\Sigma = \{+|_2, *|_2, \mathbf{1}|_0, \mathbf{2}|_0, \mathbf{3}|_0, \ldots\}$

- $\mathbf{N} = (\{1, 2, 3, \ldots\}, I_N)$ mit z.B.

  $I_N(+)(x,y) = x+y$
  $I_N(\mathbf{1}) = 1$ usw.

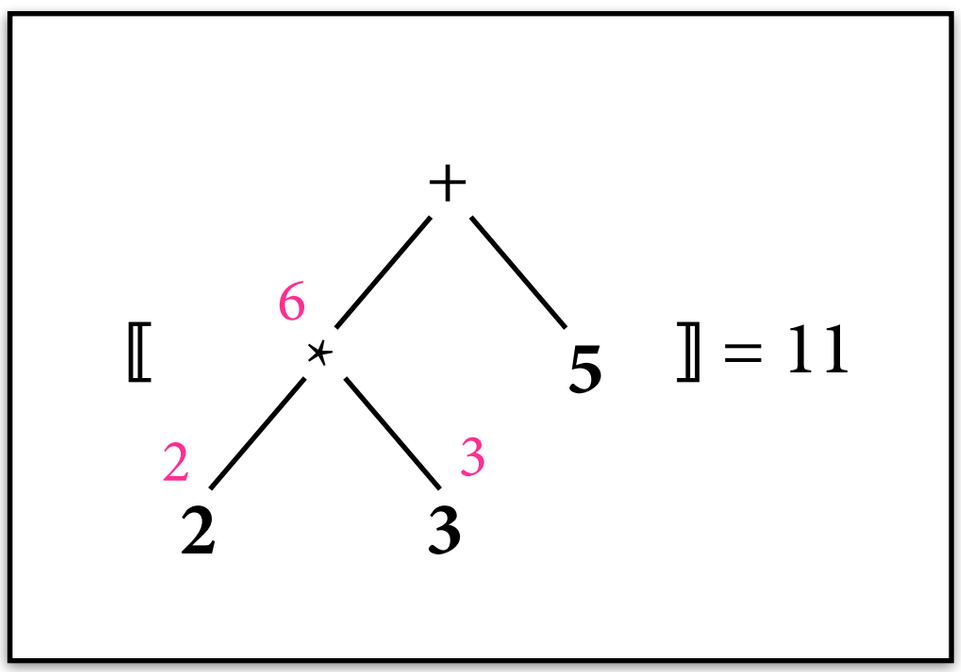


Beispiel: Arithmetik

- $\Sigma = \{+|_2, *|_2, \mathbf{1}|_0, \mathbf{2}|_0, \mathbf{3}|_0, \ldots\}$

- $\mathbf{N} = (\{1, 2, 3, \ldots\}, I_N)$ mit z.B.

  $I_N(+)(x,y) = x+y$
  $I_N(\mathbf{1}) = 1$ usw.

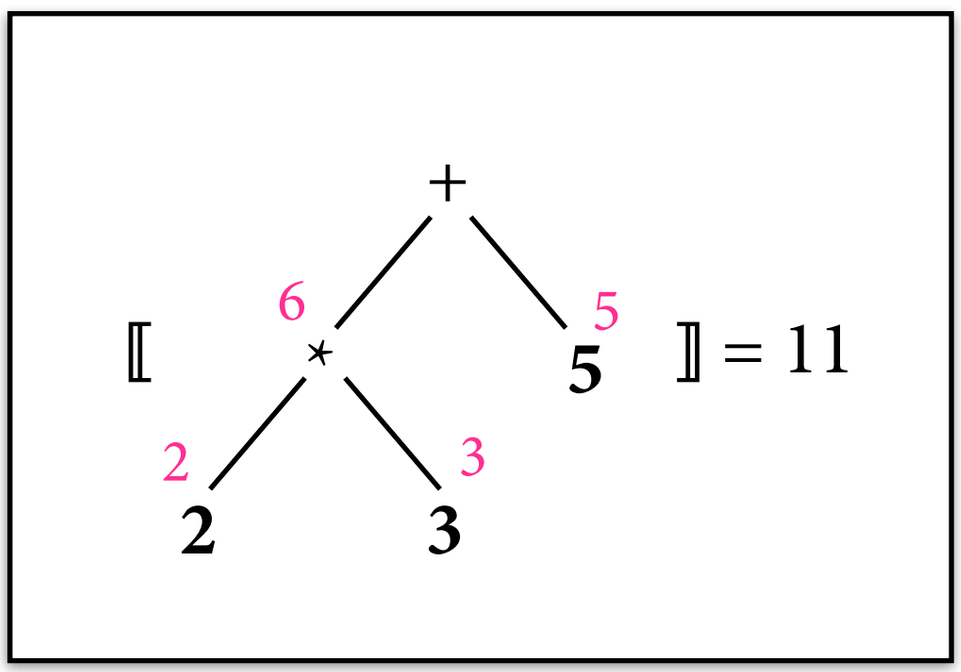


Beispiel: Arithmetik

- $\Sigma = \{+|_2, *|_2, \mathbf{1}|_0, \mathbf{2}|_0, \mathbf{3}|_0, \ldots\}$

- $\mathbf{N} = (\{1, 2, 3, \ldots\}, I_N)$ mit z.B.

  $I_N(+)(x,y) = x+y$
  $I_N(\mathbf{1}) = 1$ usw.

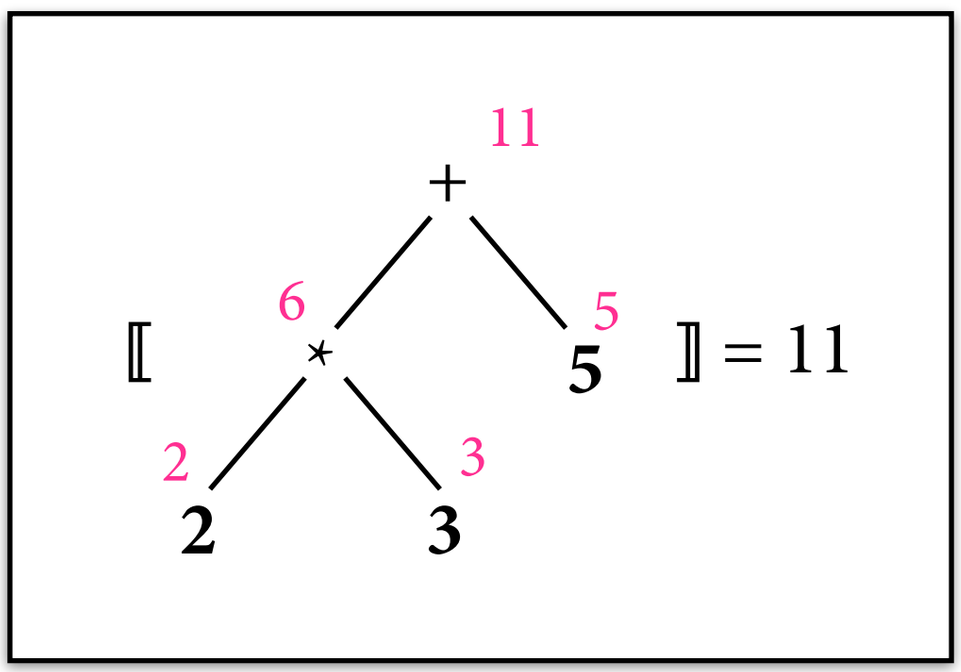


Beispiel: Arithmetik

- $\Sigma = \{+|_2, *|_2, \mathbf{1}|_0, \mathbf{2}|_0, \mathbf{3}|_0, \ldots\}$

- $\mathbf{N} = (\{1, 2, 3, \ldots\}, I_N)$ mit z.B.

  $I_N(+)(x,y) = x+y$
  $I_N(\mathbf{1}) = 1$ usw.

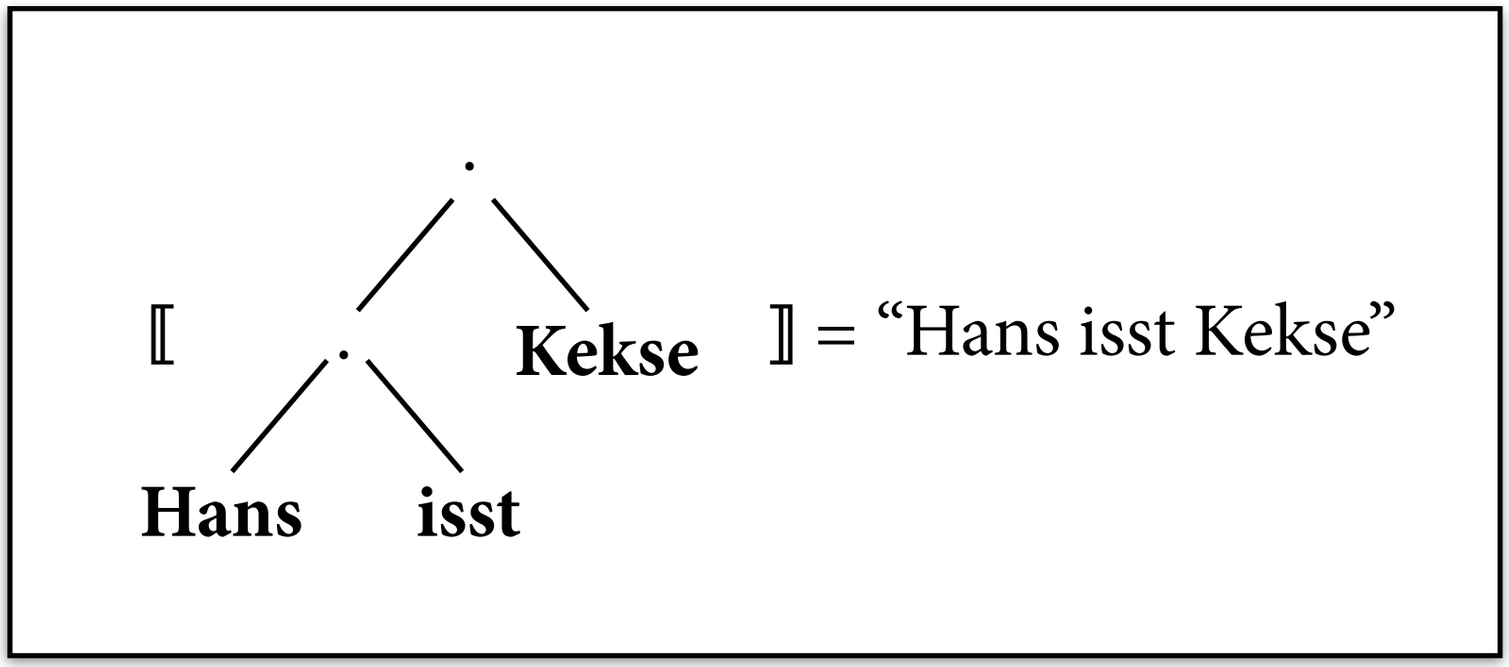


Beispiel: Strings

- $\Sigma = \{\cdot|_2, \mathbf{Hans}|_0, \mathbf{isst}|_0, \mathbf{Kekse}|_0, \ldots\}$

- $\mathbf{A} = (\{\text{Hans, isst, Hans isst, isst Kekse, } \ldots\}, I_A)$ mit z.B.

  $I_A(\cdot)(x,y) = x\ y$ (String-Konkatenation)
  $I_A(\mathbf{Hans}) =$ Hans usw.

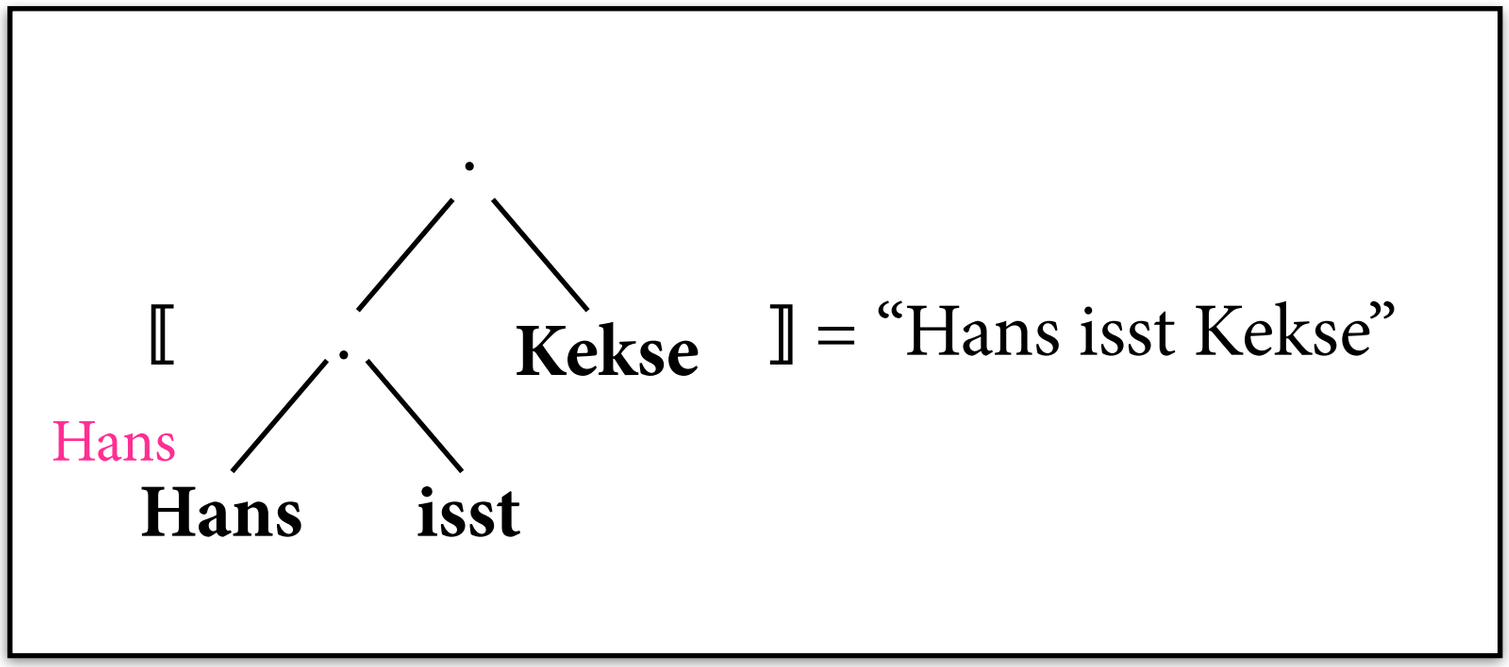


Beispiel: Strings

- $\Sigma = \{\cdot|_2, \mathbf{Hans}|_0, \mathbf{isst}|_0, \mathbf{Kekse}|_0, \ldots\}$

- $\mathbf{A} = (\{\text{Hans, isst, Hans isst, isst Kekse, } \ldots\}, I_A)$ mit z.B.

  $I_A(\cdot)(x,y) = x\ y$ (String-Konkatenation)
  $I_A(\mathbf{Hans}) =$ Hans usw.

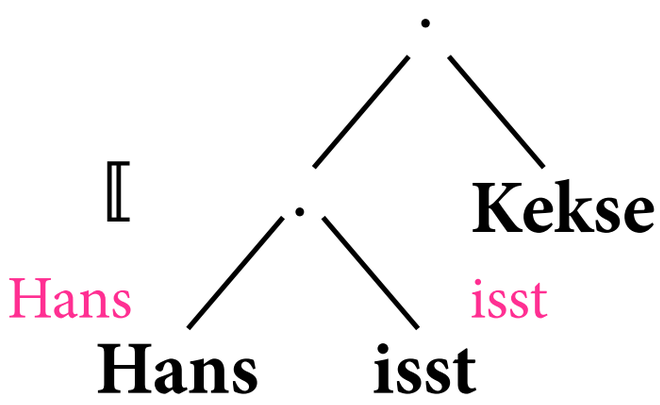


Beispiel: Strings

- $\Sigma = \{\cdot|_2, \mathbf{Hans}|_0, \mathbf{isst}|_0, \mathbf{Kekse}|_0, \ldots\}$

- $\mathbf{A} = (\{\text{Hans, isst, Hans isst, isst Kekse, } \ldots\}, I_A)$ mit z.B.

  $I_A(\cdot)(x,y) = x\ y$ (String-Konkatenation)
  $I_A(\mathbf{Hans}) =$ Hans usw.

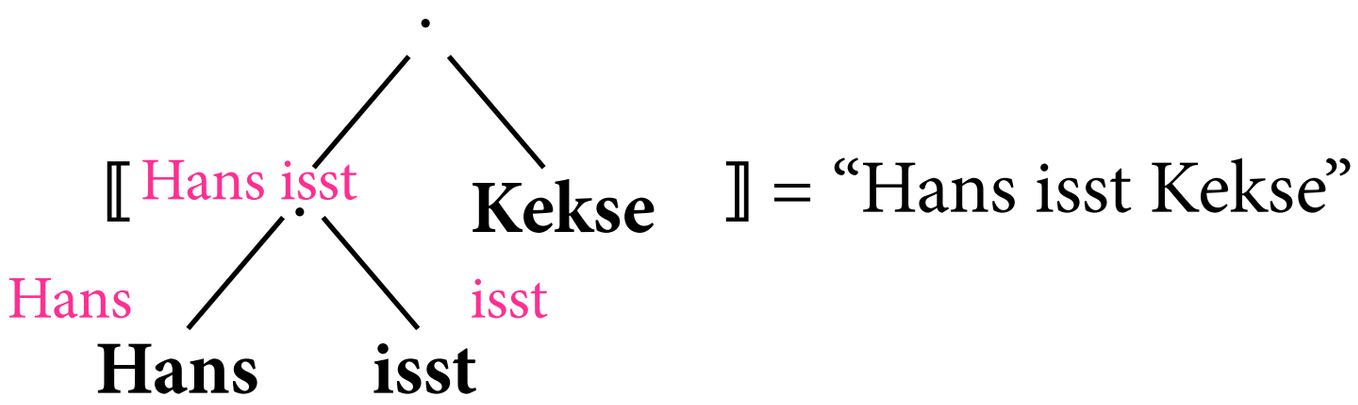


Beispiel: Strings

- $\Sigma = \{\cdot|_2, \mathbf{Hans}|_0, \mathbf{isst}|_0, \mathbf{Kekse}|_0, \ldots\}$

- $\mathbf{A} = (\{\text{Hans, isst, Hans isst, isst Kekse, } \ldots\}, I_A)$ mit z.B.

  $I_A(\cdot)(x,y) = x\ y$ (String-Konkatenation)
  $I_A(\mathbf{Hans}) =$ Hans usw.

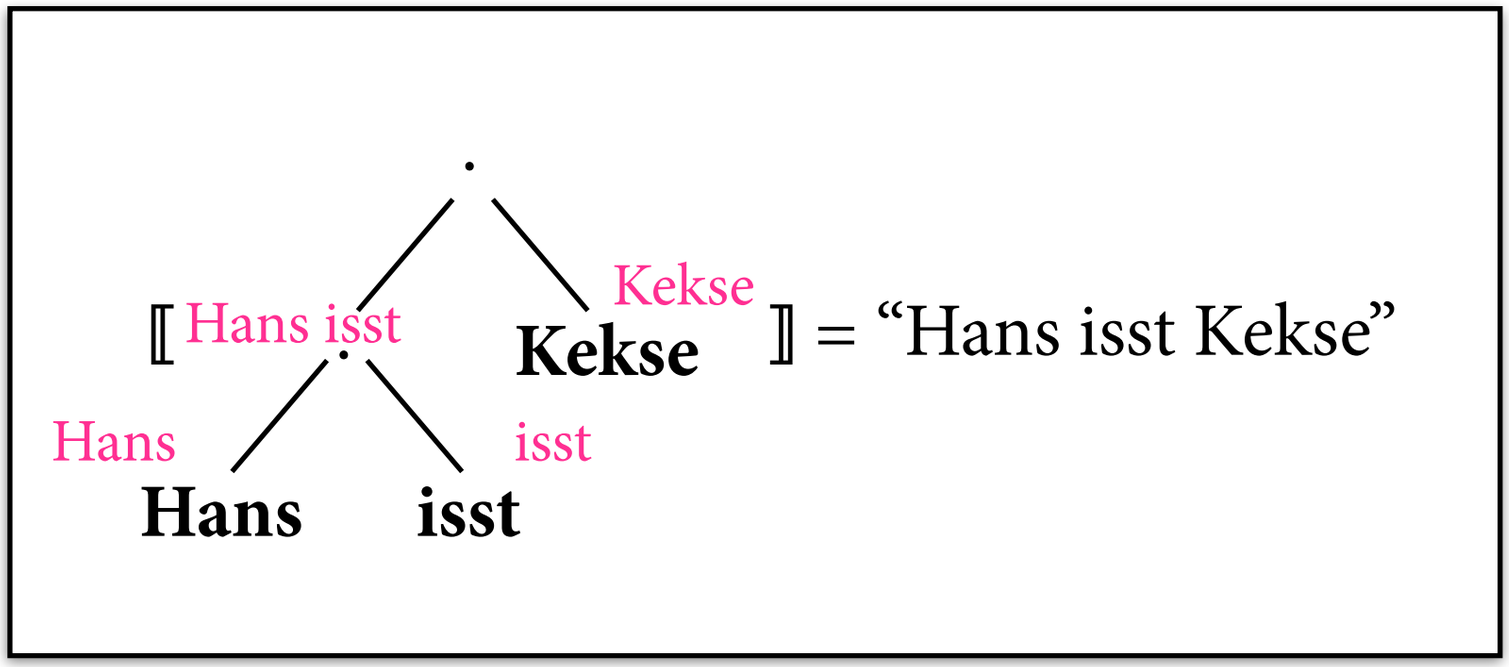


Beispiel: Strings

- $\Sigma = \{\cdot|_2, \mathbf{Hans}|_0, \mathbf{isst}|_0, \mathbf{Kekse}|_0, \ldots\}$

- $\mathbf{A} = (\{\text{Hans, isst, Hans isst, isst Kekse, } \ldots\}, I_A)$ mit z.B.

  $I_A(\cdot)(x,y) = x\ y$ (String-Konkatenation)
  $I_A(\mathbf{Hans}) =$ Hans usw.

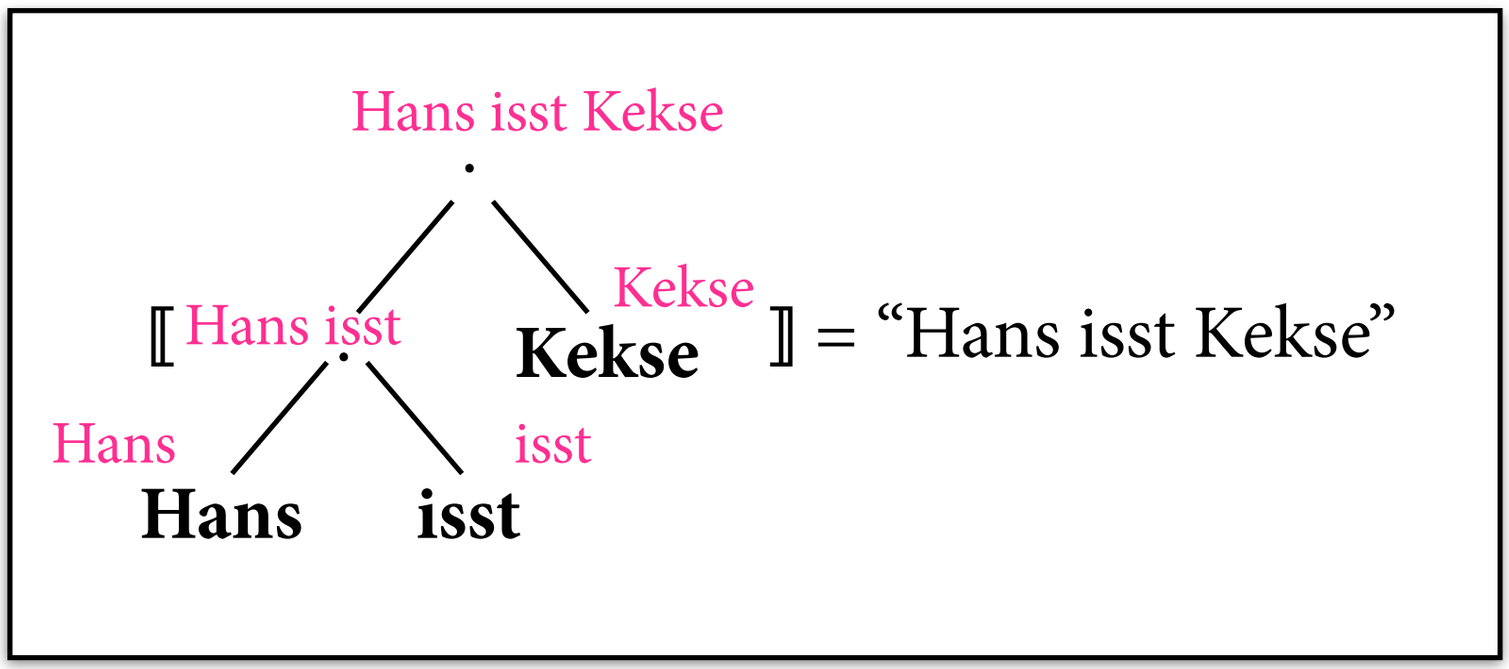


Beispiel: Strings

- $\Sigma = \{\cdot|_2, \mathbf{Hans}|_0, \mathbf{isst}|_0, \mathbf{Kekse}|_0, \ldots\}$

- $\mathbf{A} = (\{\text{Hans, isst, Hans isst, isst Kekse, } \ldots\}, I_A)$ mit z.B.

  $I_A(\cdot)(x,y) = x\ y$ (String-Konkatenation)
  $I_A(\mathbf{Hans}) =$ Hans usw.



IRTGs (vereinfacht)

- Eine interpretierte reguläre Baumgrammatik (IRTG) ist ein Paar $G = (G, (\mathbf{A}_1, \ldots, \mathbf{A}_k))$ aus:
  - einer RTG G über einer Signatur Σ
  - Algebren $\mathbf{A}_1, \ldots, \mathbf{A}_k$ über der Signatur Σ

- RTG beschreibt eine Sprache $L(G) \subseteq T_\Sigma$ von *Ableitungsbäumen*.

- IRTG beschreibt die Sprache
  $L(\mathbf{G}) = \{ (\llbracket t \rrbracket_{\mathbf{A1}}, \ldots, \llbracket t \rrbracket_{\mathbf{Ak}}) \mid t \in L(G) \}$
  $\subseteq A_1 \times \ldots \times A_k$

Beispiel

G = (G, **A**) mit **A** = Stringalgebra

G

S → ·(NP, VP)

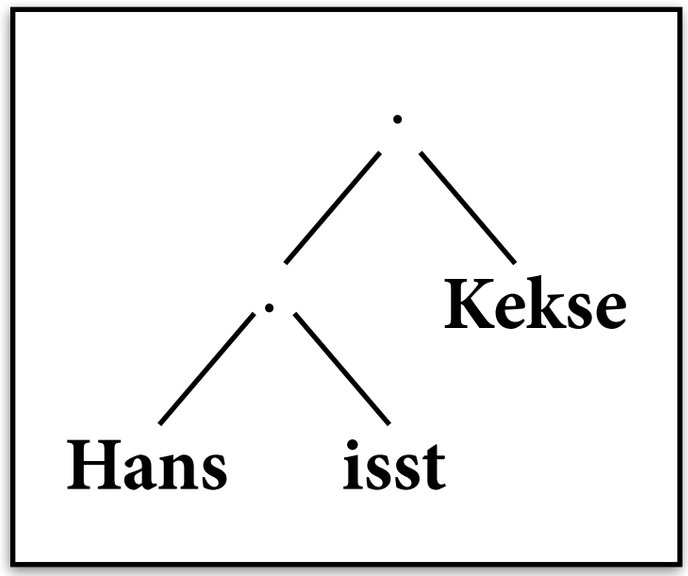
VP → ·(V, NP)

NP → **Hans**

V → **isst**

NP → **Kekse**

Abl.baum t ∈ L(G)



"Hans isst Kekse"

Wert ⟦t⟧ ∈ A

⇒ "Hans isst Kekse" ∈ L(**G**)

Parsing mit IRTGs

- Sei $\mathbf{G} = (G, (\mathbf{A}_1, \ldots, \mathbf{A}_k))$ IRTG und $a \in A_1$ Eingabe.

- Gesucht: $\text{parses}(a) = \{ t \in L(G) \mid [\![t]\!]_{A1} = a \}$
  $= L(G) \cap \text{terms}(a)$

  mit $\text{terms}(a) = \{ t \in T_\Sigma \mid [\![t]\!]_{A1} = a \}$.

- Nenne $\mathbf{A}_1$ *regulär zerlegbar*, wenn für jedes $a \in A_1$ eine RTG D_a existiert mit $L(D_a) = \text{terms}(a)$ *(Zerlegungsgrammatik)*.

- Dann gilt: $\text{parses}(a) = L(G) \cap L(D_a)$.

Beispiel

S → ·(NP, VP)

VP → ·(V, NP)

NP → **Hans**

V → **isst**

NP → **Kekse**

G

S[0,3] → ·(NP[0,1], VP[1,3])

VP[1,3] → ·(V[1,2], NP[2,3])

NP[0,1] → **Hans**

V[1,2] → **isst**

NP[2,3] → **Kekse**

$G \cap D_a$

Zerlegungsgrammatik D_a

[0,3] → ·([0,1], [1,3])

[0,3] → ·([0,2], [2,3])

[1,3] → ·([1,2], [2,3])

[0,1] → **Hans**

[1,2] → **isst**

[2,3] → **Kekse**

Eingabe a ∈ A

“Hans isst Kekse”

Baum-Homomorphismen

- Einfache Abbildungen von Bäumen über Signatur Σ in Bäume über Signatur Δ.

- Definiere Bild für jedes Symbol $f|n \in \Sigma$ als Term über Δ mit Variablen $x_1, \ldots, x_n$: $h_f \in T_\Delta[x_1,\ldots,x_n]$.

- Dann rekursiv auf Funktion $h: T_\Sigma \rightarrow T_\Delta$ fortsetzen: $h(f(t_1, \ldots, t_n)) = h_f [h(t_1)/x_1, \ldots, h(t_n)/x_n]$

IRTGs (vollständige Version)

- Eine interpretierte reguläre Baumgrammatik (IRTG) ist ein Paar $G = (G, ((h_1,\mathbf{A}_1), \ldots, (h_k,\mathbf{A}_k)))$ aus:
  - einer RTG G über einer Signatur Σ
  - Algebren $\mathbf{A}_1, \ldots, \mathbf{A}_k$ über Signaturen $\Delta_1, \ldots, \Delta_k$
  - Baumhomomorphismen $h_i: T_\Sigma \to T_{\Delta i}$

- IRTG beschreibt die Sprache

$$L(\mathbf{G}) = \{ (\llbracket h_1(t) \rrbracket_{\mathbf{A1}}, \ldots, \llbracket h_k(t) \rrbracket_{\mathbf{Ak}}) \mid t \in L(G) \}$$
$$\subseteq A_1 \times \ldots \times A_k$$

Beispiel

$S \to s_2(NP, VP)$

$VP \to vp_2(V, NP)$

$NP \to hans_0$

$V \to isst_0$

$NP \to kekse_0$

G

$h(s_2) = \cdot(x_1, x_2)$

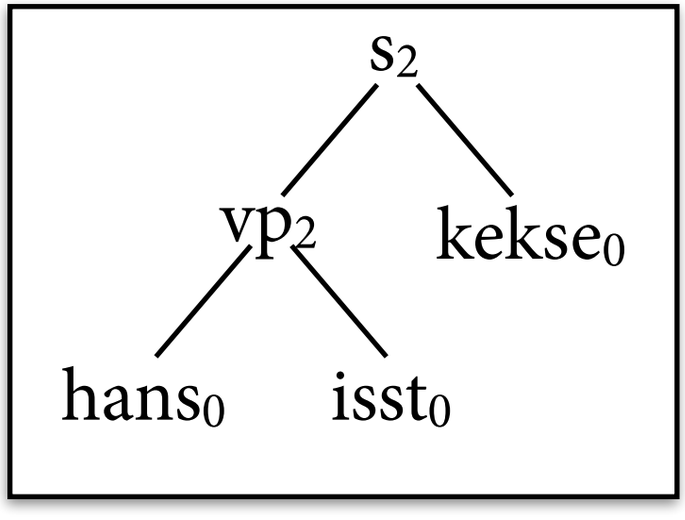
$h(vp_2) = \cdot(x_1, x_2)$

$h(hans_0) = \mathbf{Hans}$

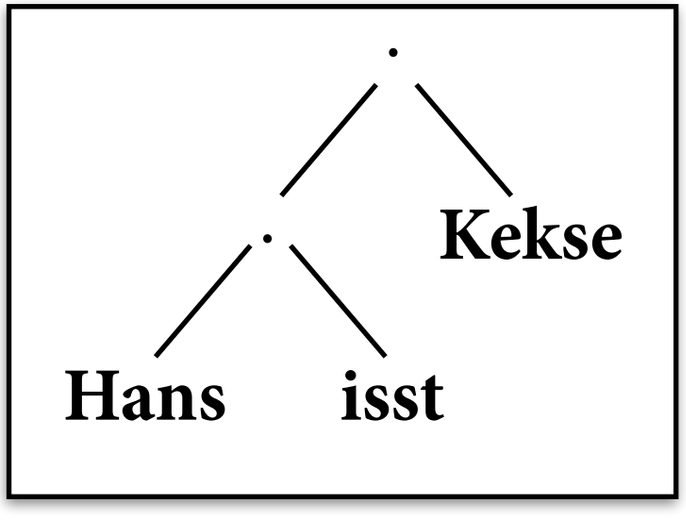
$h(isst_0) = \mathbf{isst}$

$h(kekse_0) = \mathbf{Kekse}$

Abl.baum $t \in L(G)$



Term $h(t) \in T_\Delta$



Wert $[\![h(t)]\!] \in A$

“Hans isst Kekse”

IRTGs: Ausblick

- Parsing: funktioniert auch mit Homomorphismen: $\text{parses}(a) = L(G) \cap h^{-1}(L(D_a))$

- Mehrere Interpretationen ($k > 1$): Relationen zwischen Strings, Bäumen usw.
  - maschinelle Übersetzung
  - Abbildung von String in Parsebäume oder sem. Repräsentationen, oder umgekehrt

- PCFG-artige und log-lineare W.modelle sehr einfach zu übertragen.

Zusammenfassung

- Reguläre Baumgrammatiken / -sprachen und endliche Baumautomaten.
  - Abschlusseigenschaften!

- Interpretierte RTGs:
  - Beschreibe Sprache von Objekten in beliebiger Algebra.
  - Regulär zerlegbare Algebra ⇒ Parsing mit RTGs.
  - Implementierung: http://bitbucket.org/tclup/alto