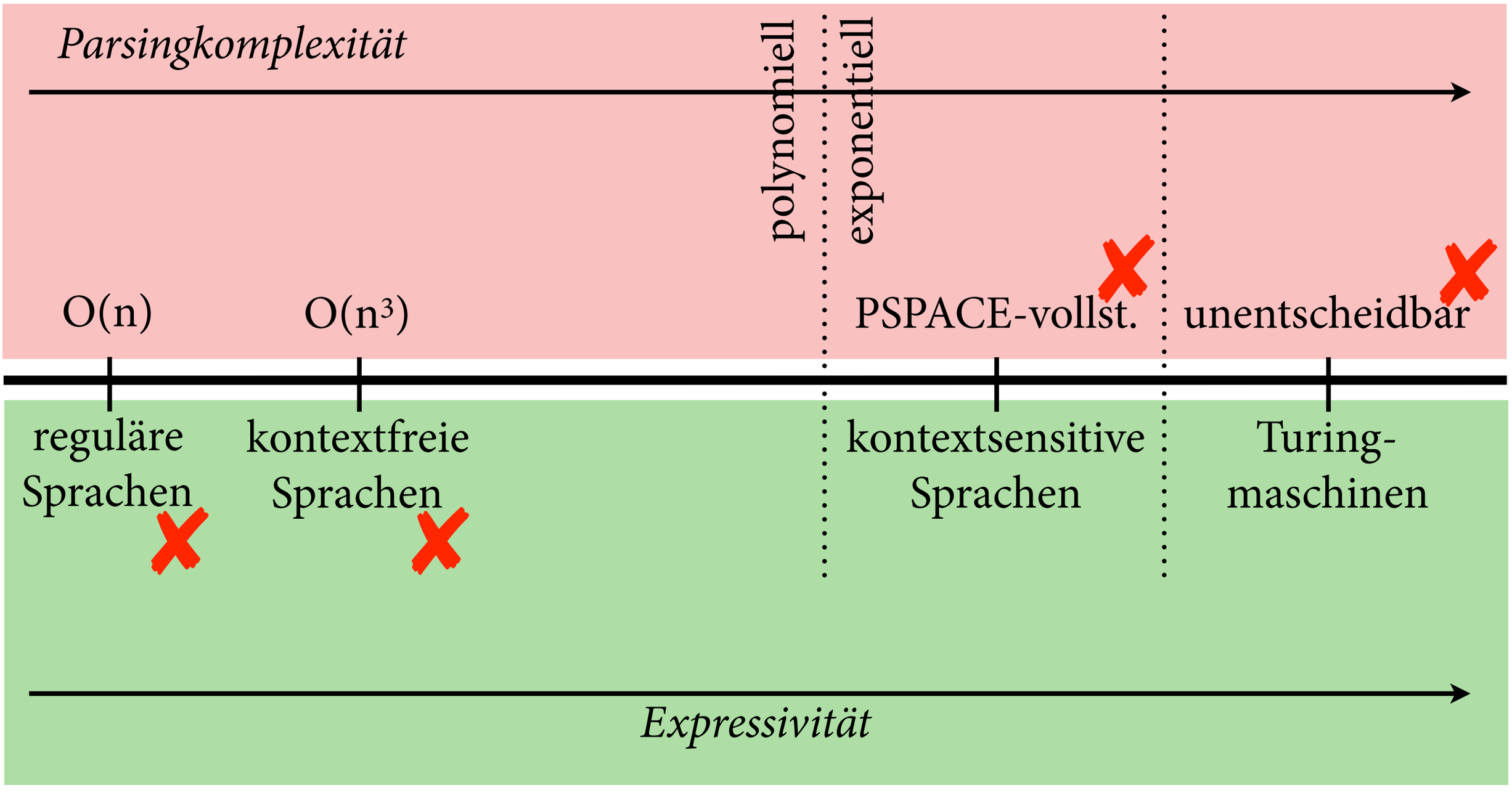
Schwach kontextsensitive Grammatikformalismen

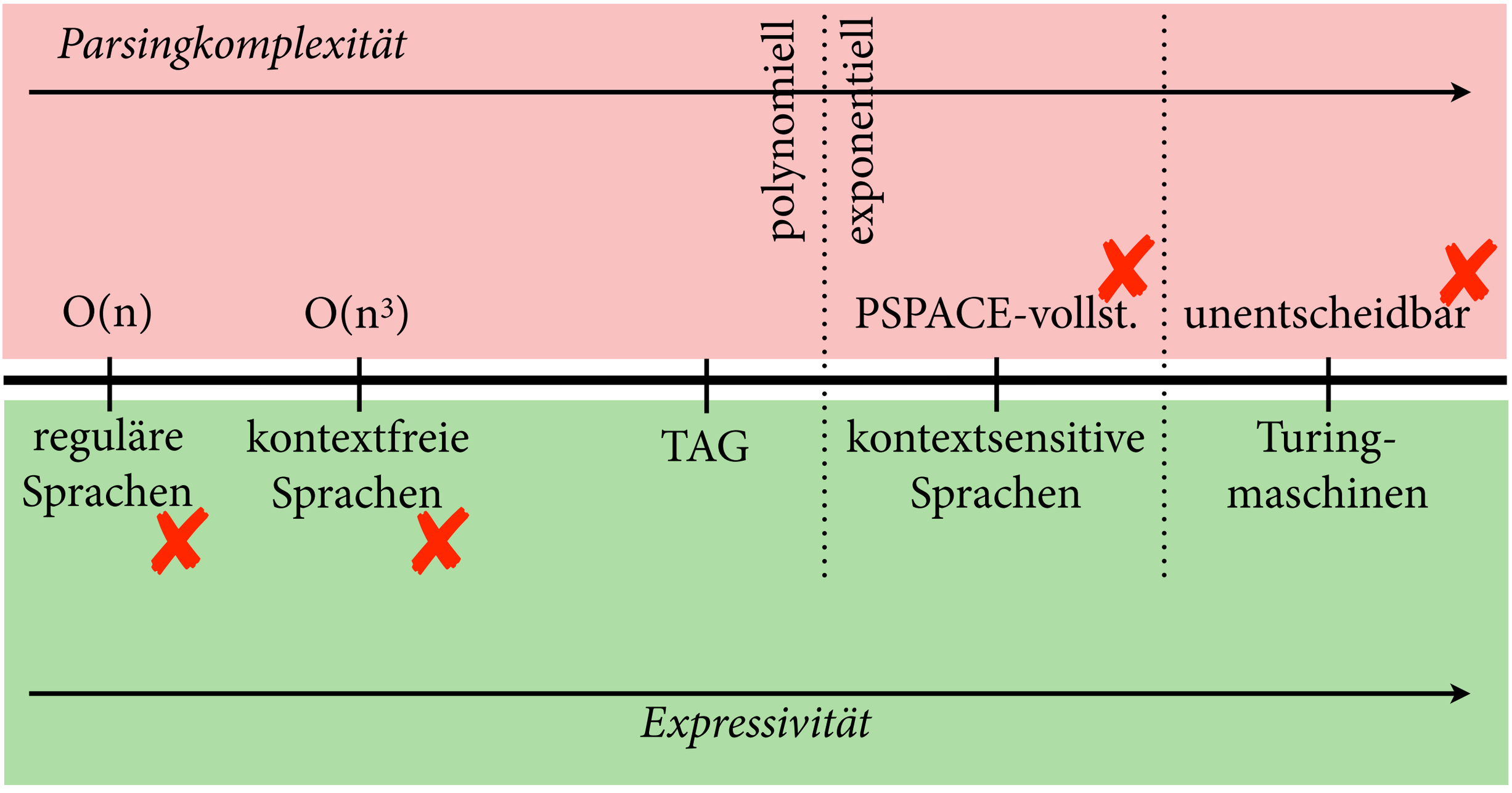
Vorlesung “Grammatikformalismen”
Alexander Koller

24. Mai 2019

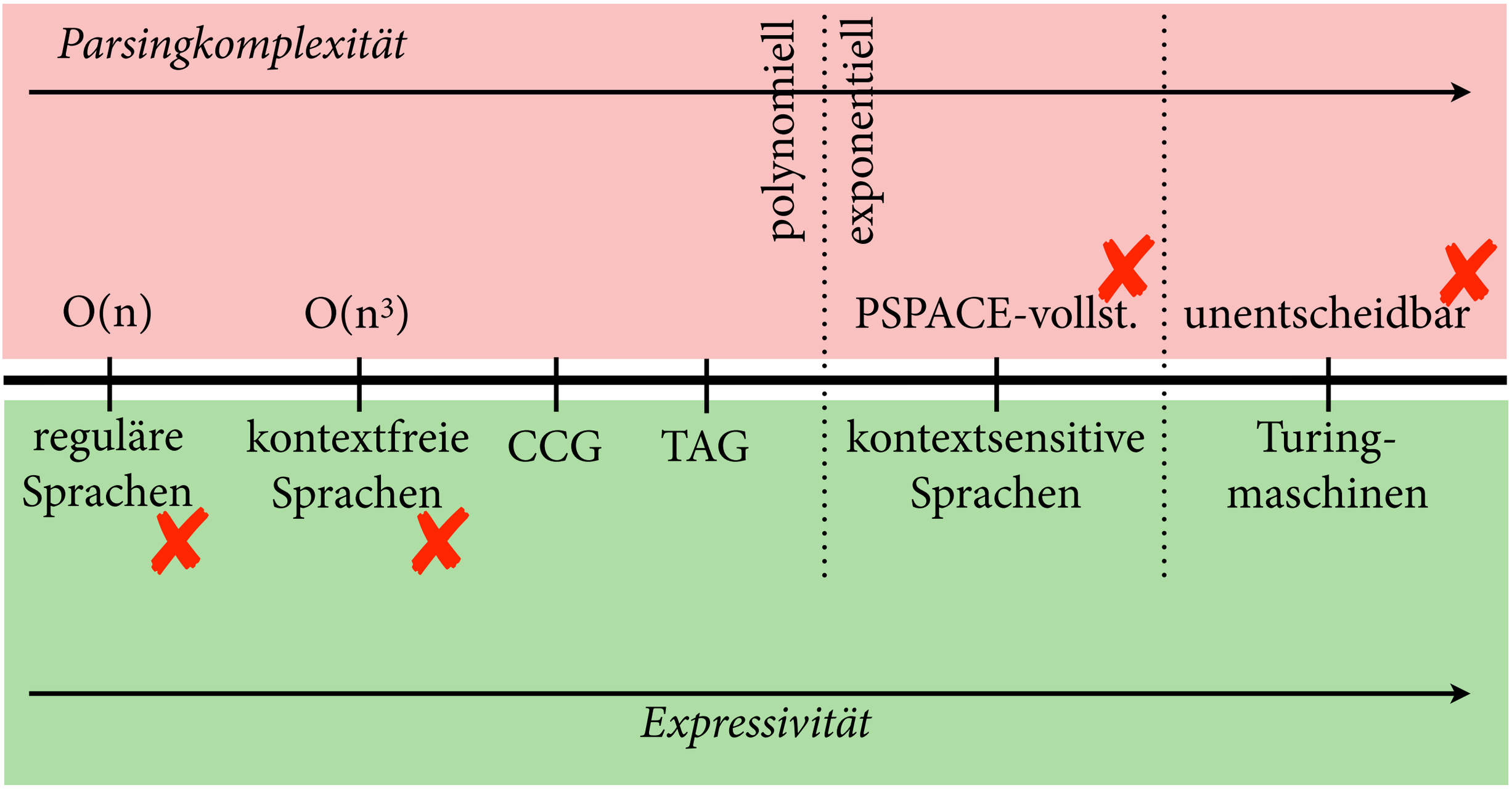
Grammatikformalismen



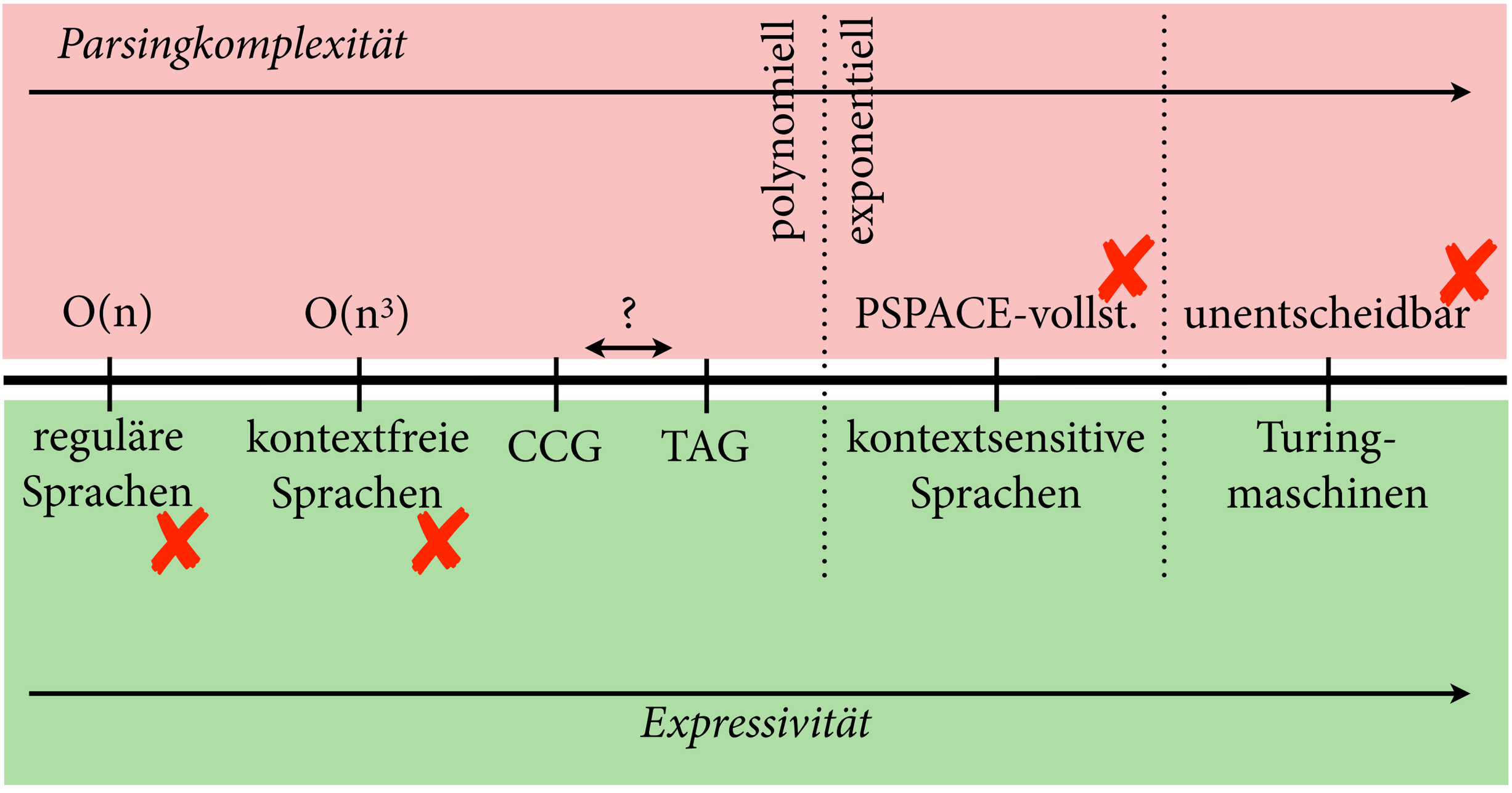
Grammatikformalismen



Grammatikformalismen



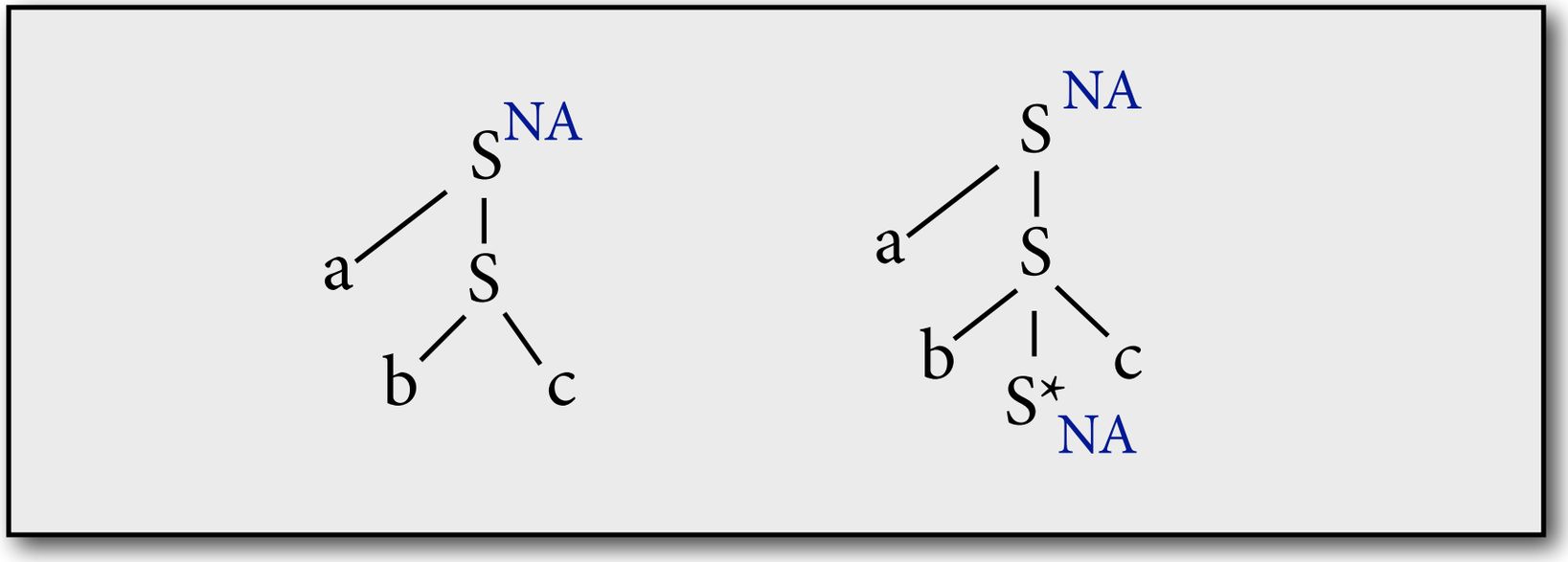
Grammatikformalismen

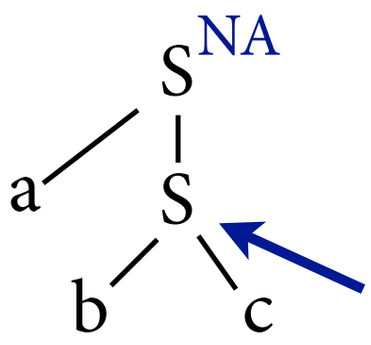


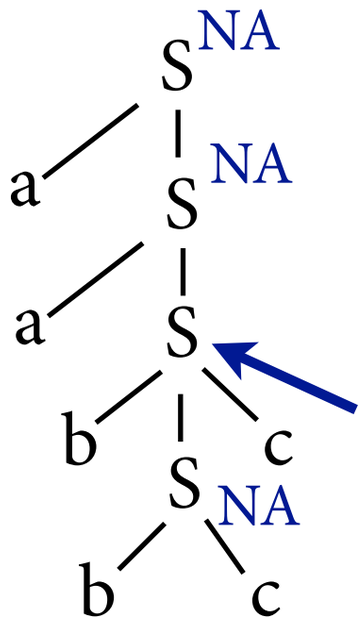
Die Sprache COUNT(3)

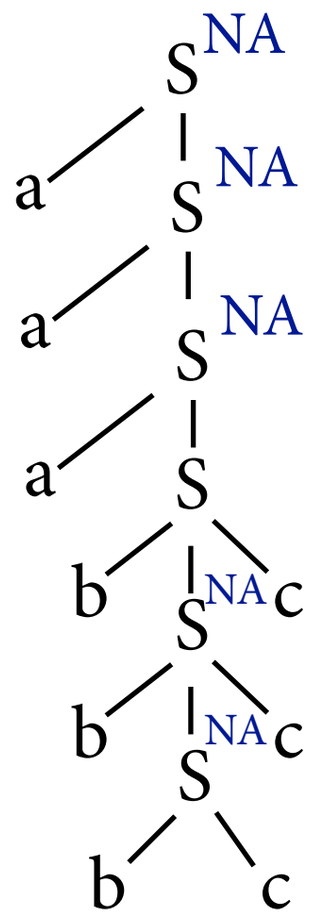
- COUNT(3) = $a^n\ b^n\ c^n$
- Wir wissen, dass COUNT(3) keine kontextfreie Sprache ist.
  - z.B. mit dem Pumping-Lemma beweisen
- Mit TAG geht es.
- Mit CCG geht es auch.

TAG für COUNT(3)







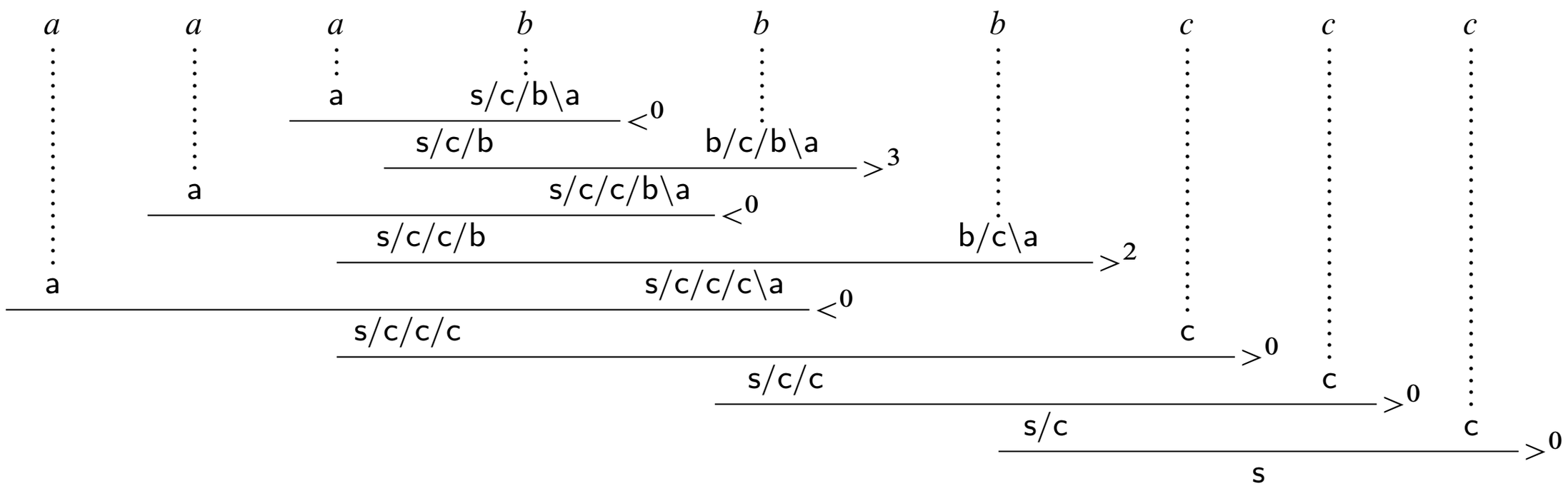


CCG für COUNT(3)

Lexikon:

a: A	c: C	b: S/C/B\A
b: S/C\A	b: B/C/B\A	b: B/C\A

Sekundäre Prämisse jeder Kompositionsregel muss von der Form B/C/B\A oder B/C\A sein.



TAG und CCG

- Beide können COUNT(3) und COUNT(4).
- Keiner der beiden kann COUNT(5).
- Wortproblem in beiden Fällen $O(n^6)$.
- Kann das alles Zufall sein?

Übersicht

- Äquivalenz von TAG und CCG
- Schwach kontextsensitive Grammatikformalismen
- Reguläre Baumsprachen

Äquivalenz von Grammatiken

- Äquivalenzbegriffe für kfGen:
  - Grammatiken G_1 und G_2 sind *schwach äquivalent*, wenn $L(G_1) = L(G_2)$, d.h. gleiche Sprache.
  - G_1 und G_2 sind *stark äquivalent*, wenn sie jedem String die gleichen Parsebäume zuweisen.

- Wie vergleicht man Äquivalenz über Formalismusgrenzen hinweg?

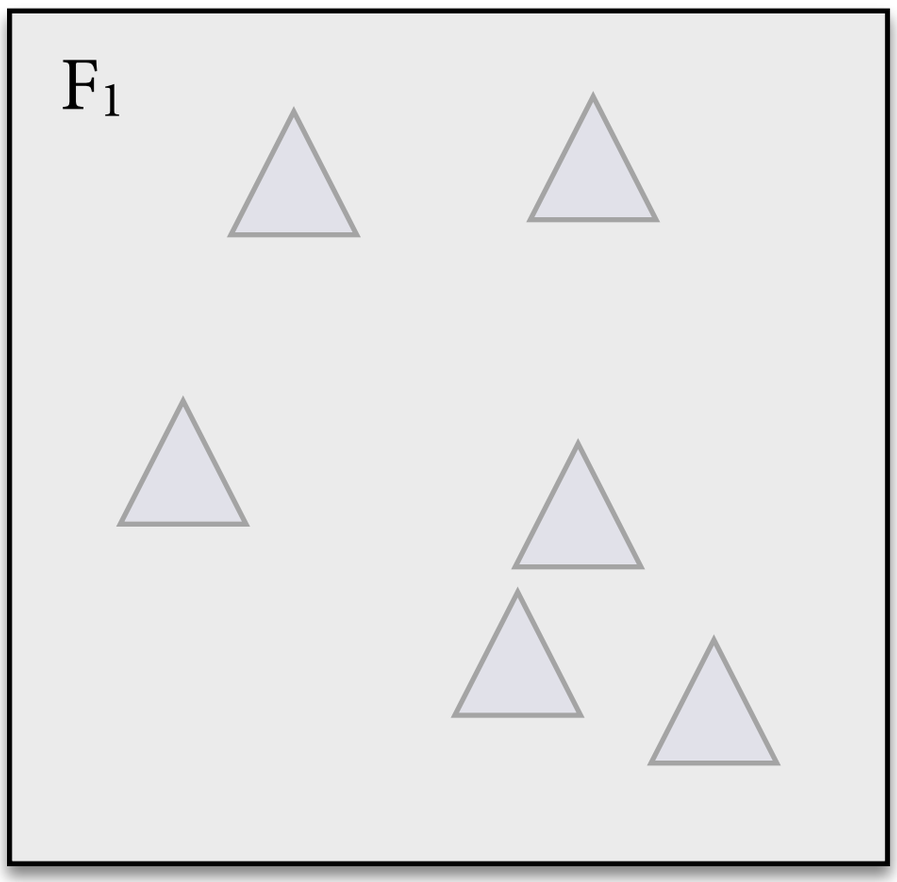
Äquivalenz von Grammatiken

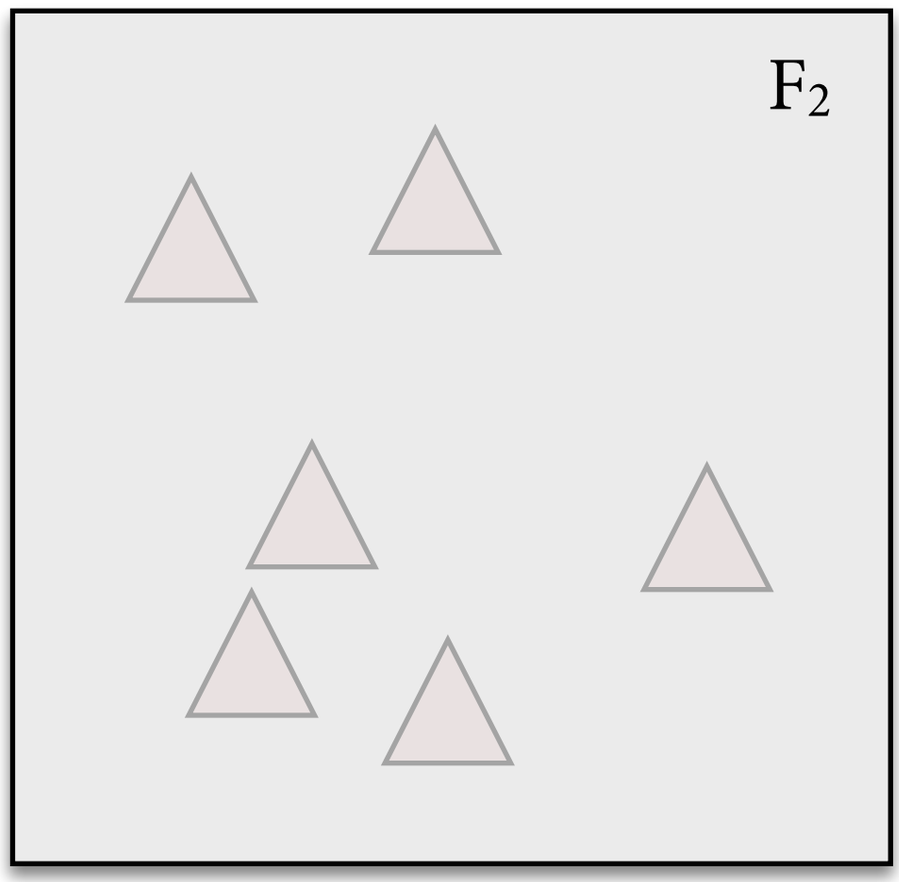
- Seien:
  - F_1 ein Grammatikformalismus; T_1 alle in F_1 erlaubten syntaktischen Ableitungen (Parsebäume, o.ä.)
  - F_2 ein Grammatikformalismus mit Ableitungen T_2
  - G_1 eine Grammatik von F_1, die Menge $T(G_1)$ als grammatisch korrekt auszeichnet; G_2 Grammatik von F_2 mit grammatisch korrekten Ableitungen $T(G_2)$
  - V eine Menge von *Vergleichsstrukturen* mit Abbildungen $i_1: T_1 \rightarrow V$, $i_2: T_2 \rightarrow V$

- G_1 und G_2 heißen *äquivalent* bezüglich i_1 und i_2, wenn $i_1(T(G_1)) = i_2(T(G_2))$.

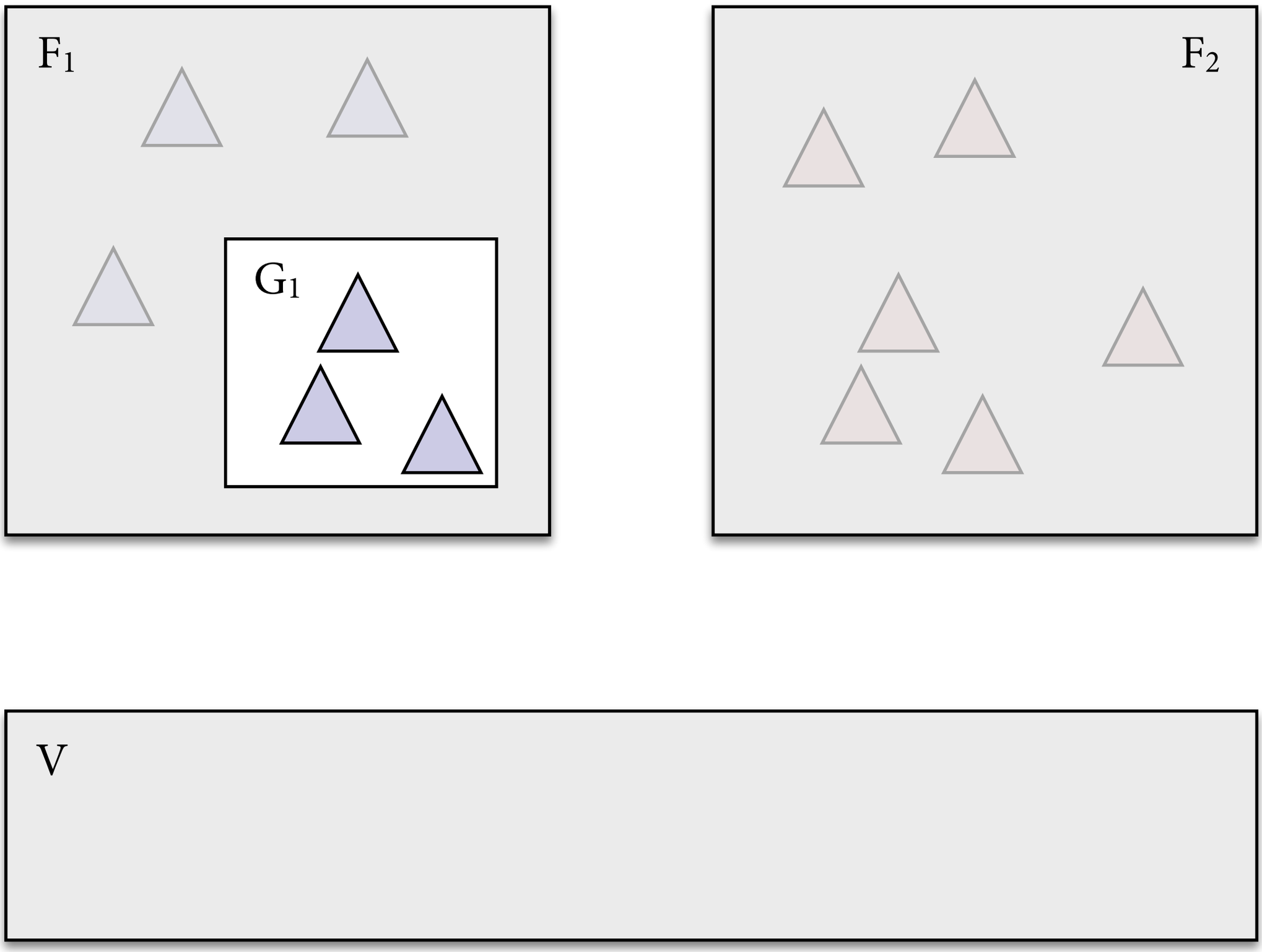
(Miller 99)

Äquivalenz von Grammatiken

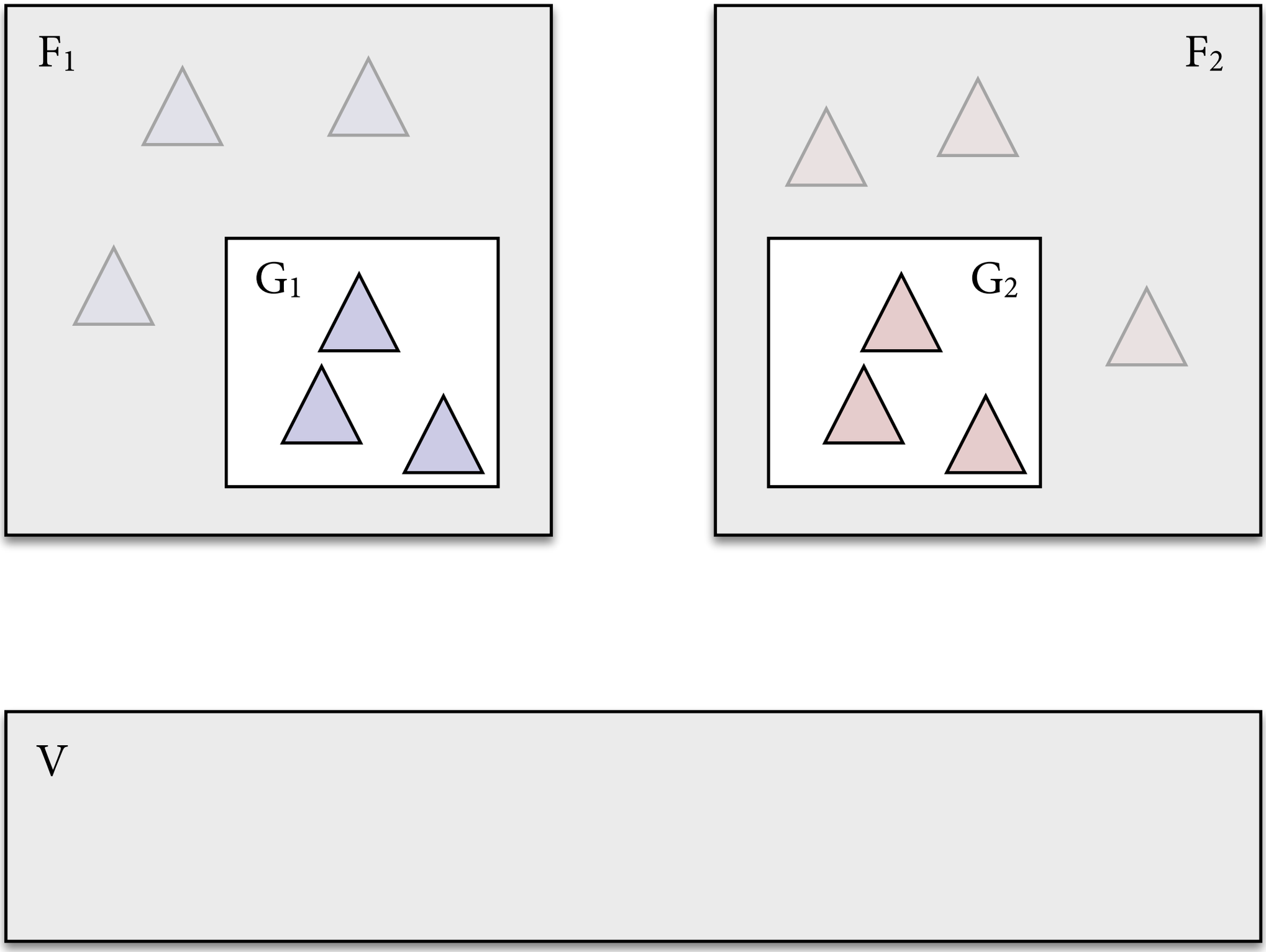




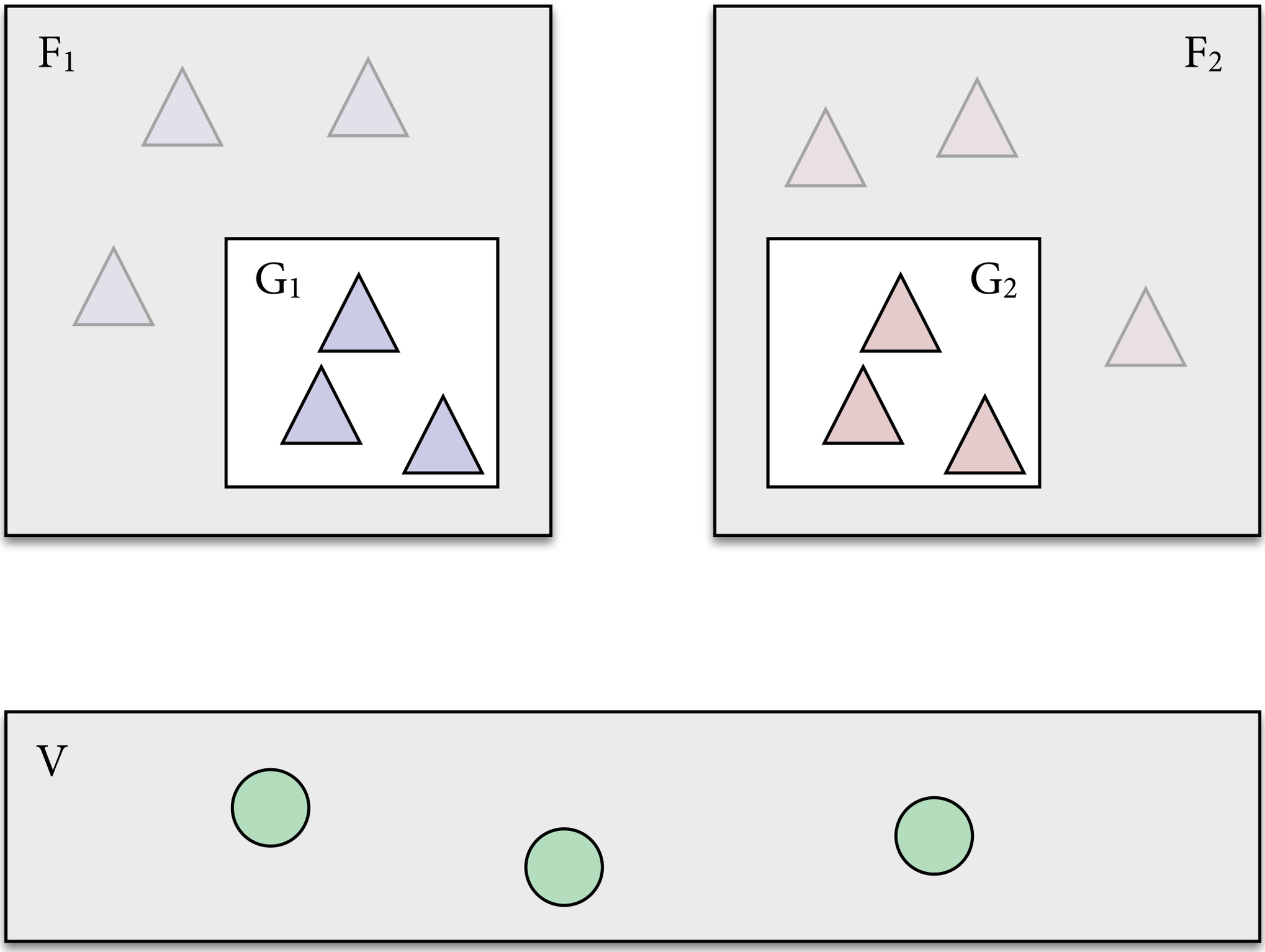
Äquivalenz von Grammatiken



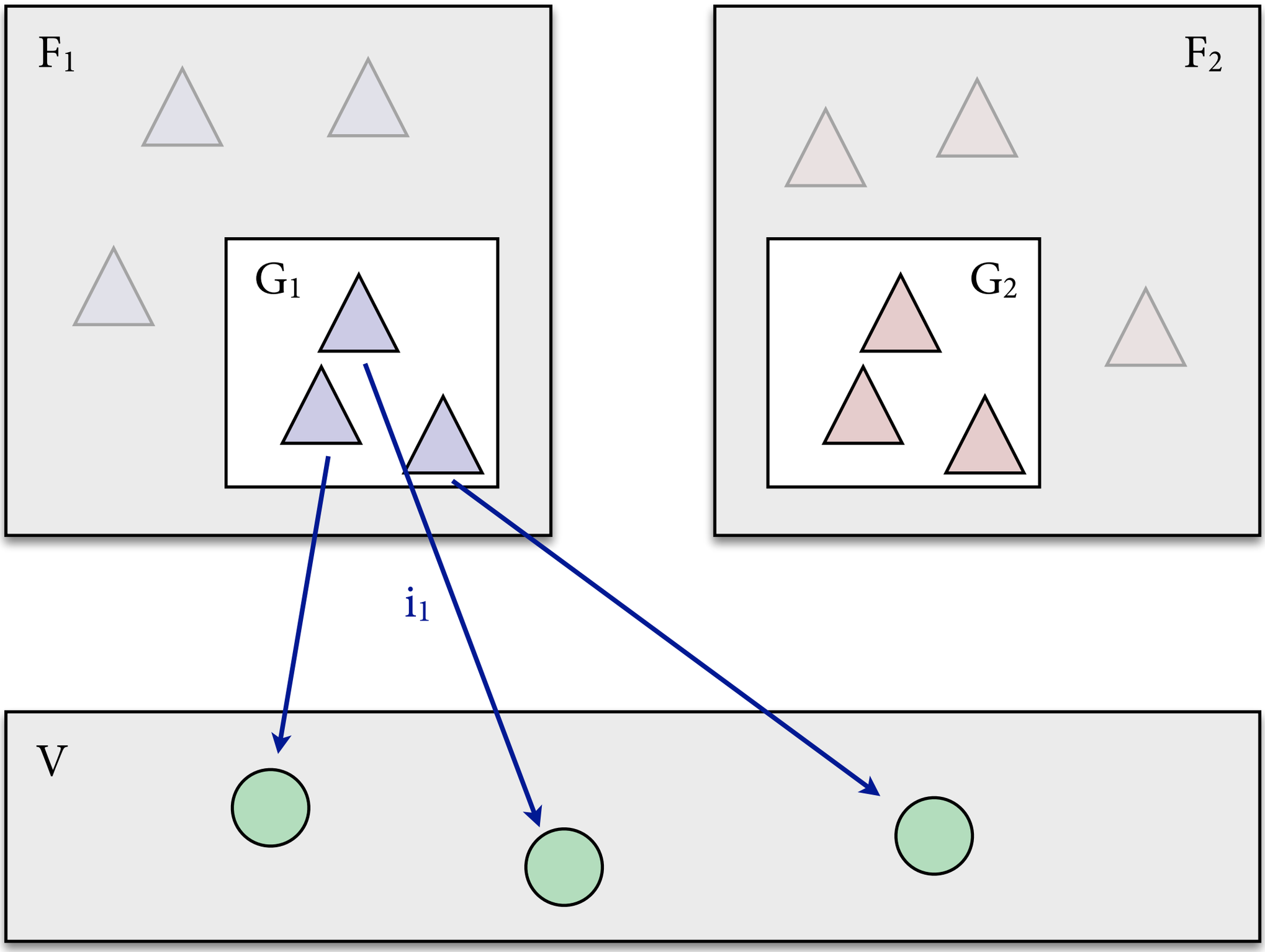
Äquivalenz von Grammatiken



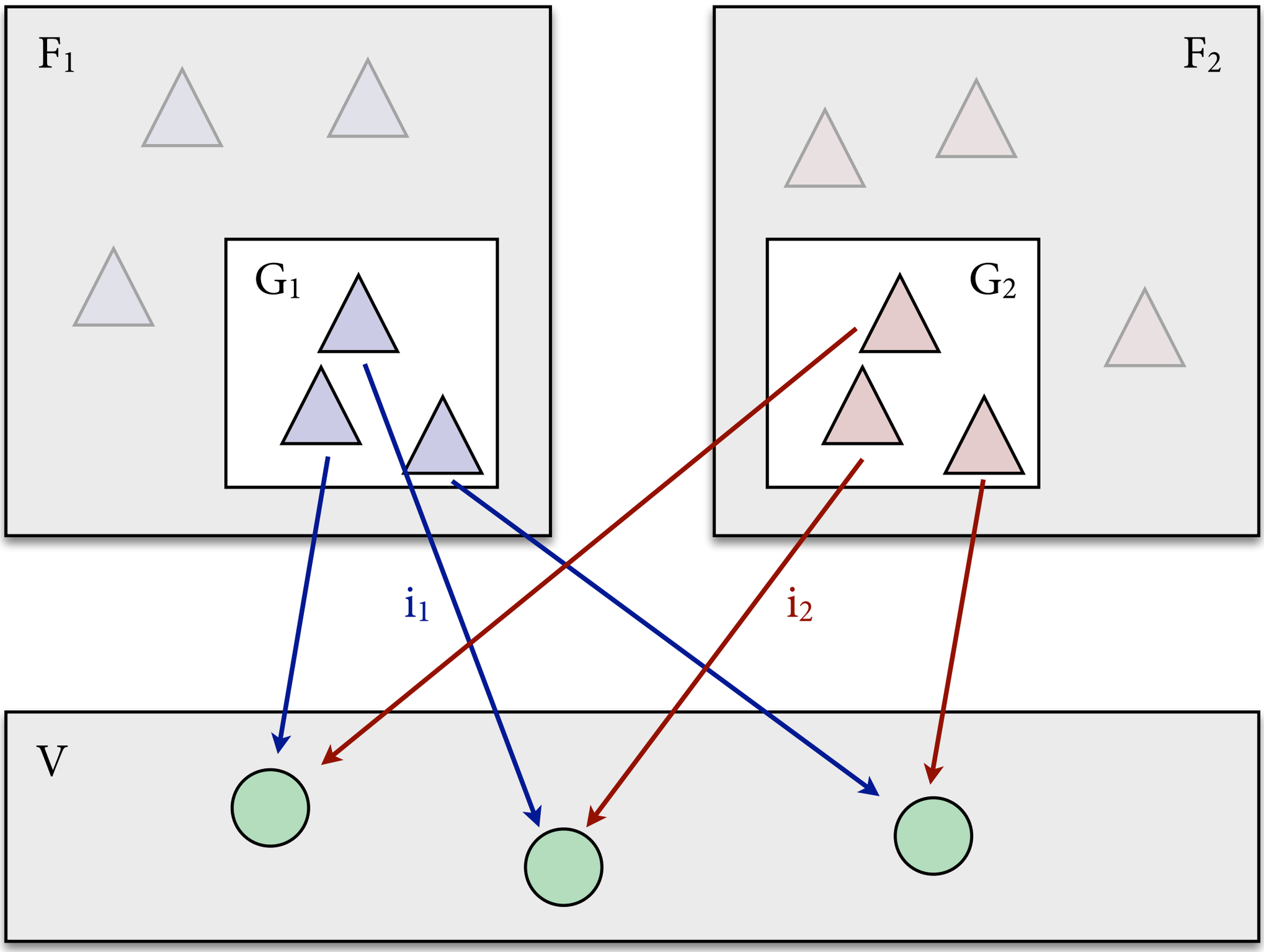
Äquivalenz von Grammatiken



Äquivalenz von Grammatiken



Äquivalenz von Grammatiken



Beispiele

- *Schwache Äquivalenz:* V = Strings,
  i_1 und i_2 bilden Ableitung auf geparsten String ab.
  - auch wenn G_1 und G_2 zu verschiedenen Grammatikformalismen gehören

- *Starke Äquivalenz* im kfG-Sinn:
  V = kontextfreie Parsebäume,
  i_1 und i_2 identische Abbildungen.

- Dazwischen viele Alternativen denkbar.

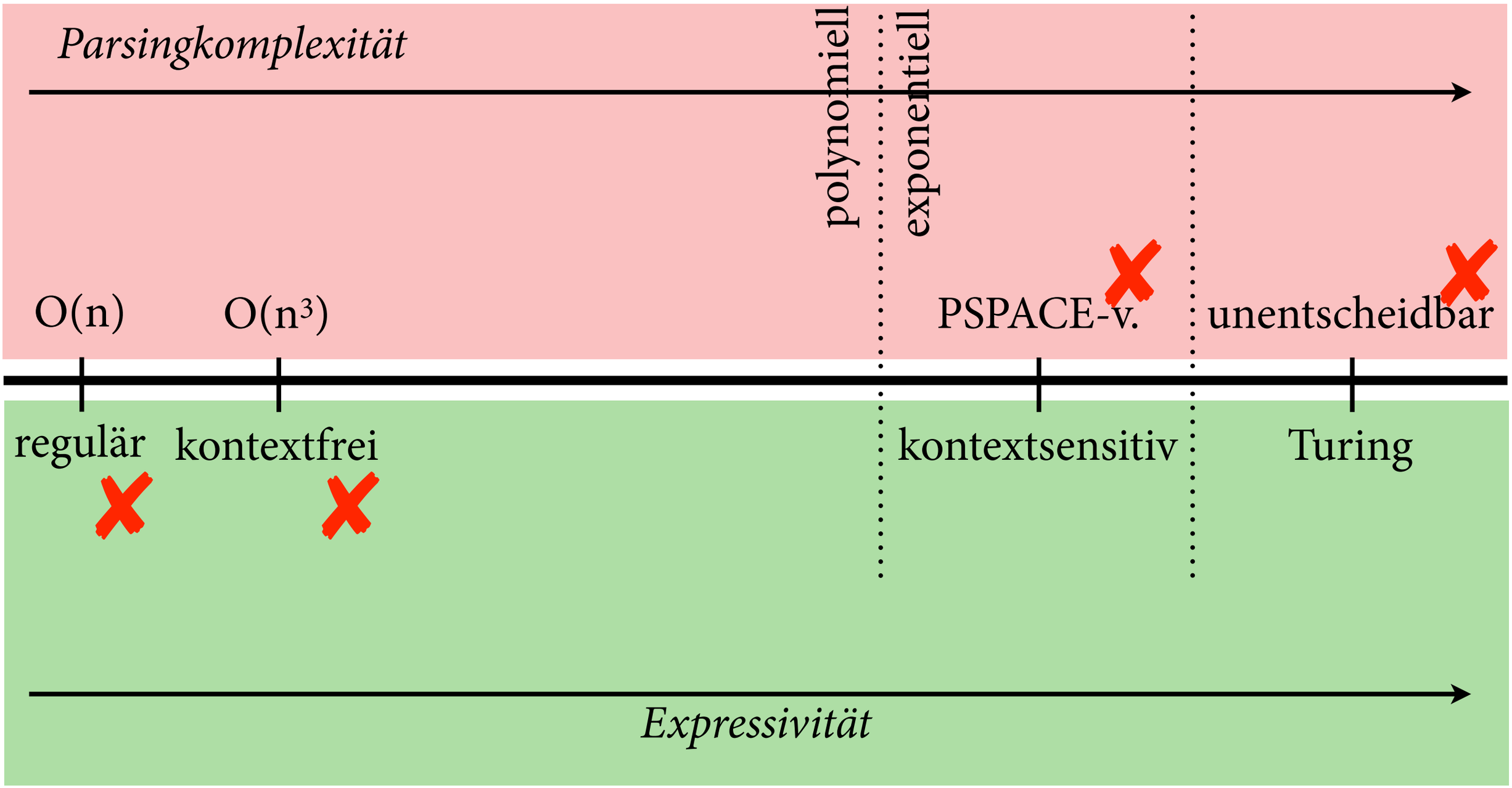
Äquivalenz von TAG und CCG

- Die folgenden Formalismen sind schwach äquivalent:
  - TAG (mit Adjunktionsconstraints)
  - CCG (mit Regeleinschränkungen)
  - Head Grammars (Pollard)
  - Linear Indexed Grammars (Gazdar)

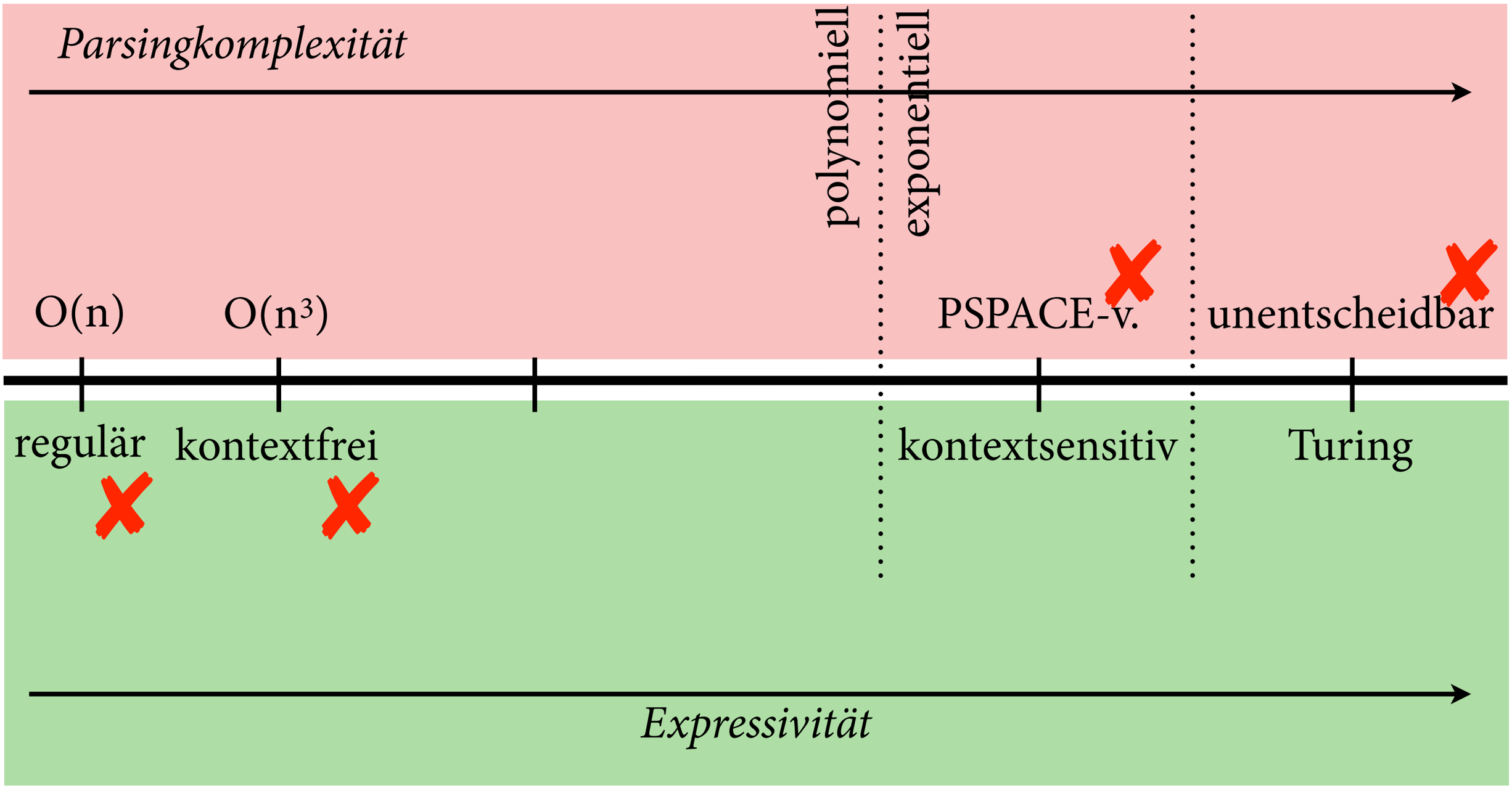
- Beweis ist konstruktiv, aber etwas indirekt.
  - Bei Umformung von Grammatiken in Normalformen wird Struktur des Ableitungsbaums zerstört.

(Vijay-Shanker & Weir 1994)

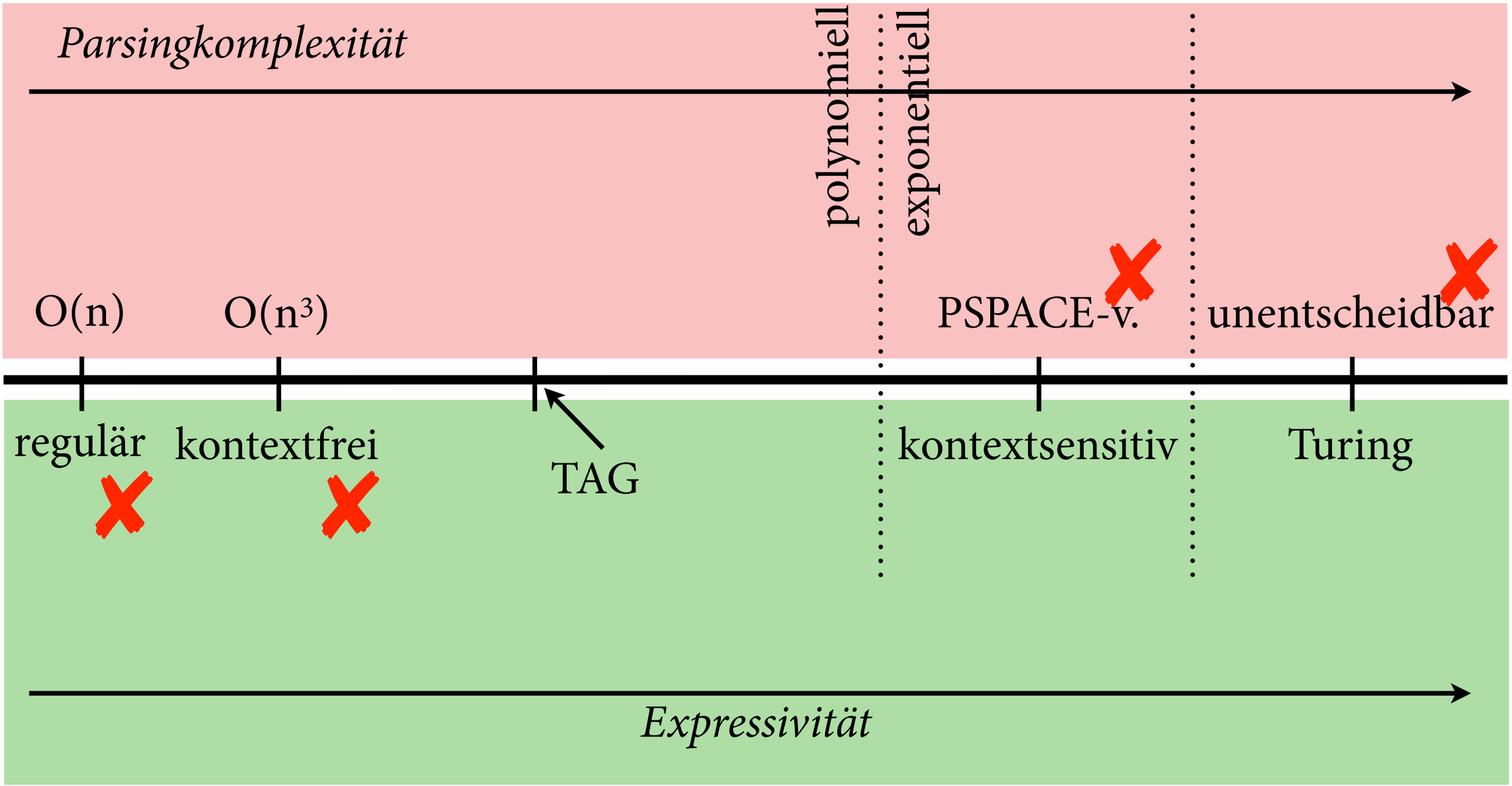
Grammatikformalismen



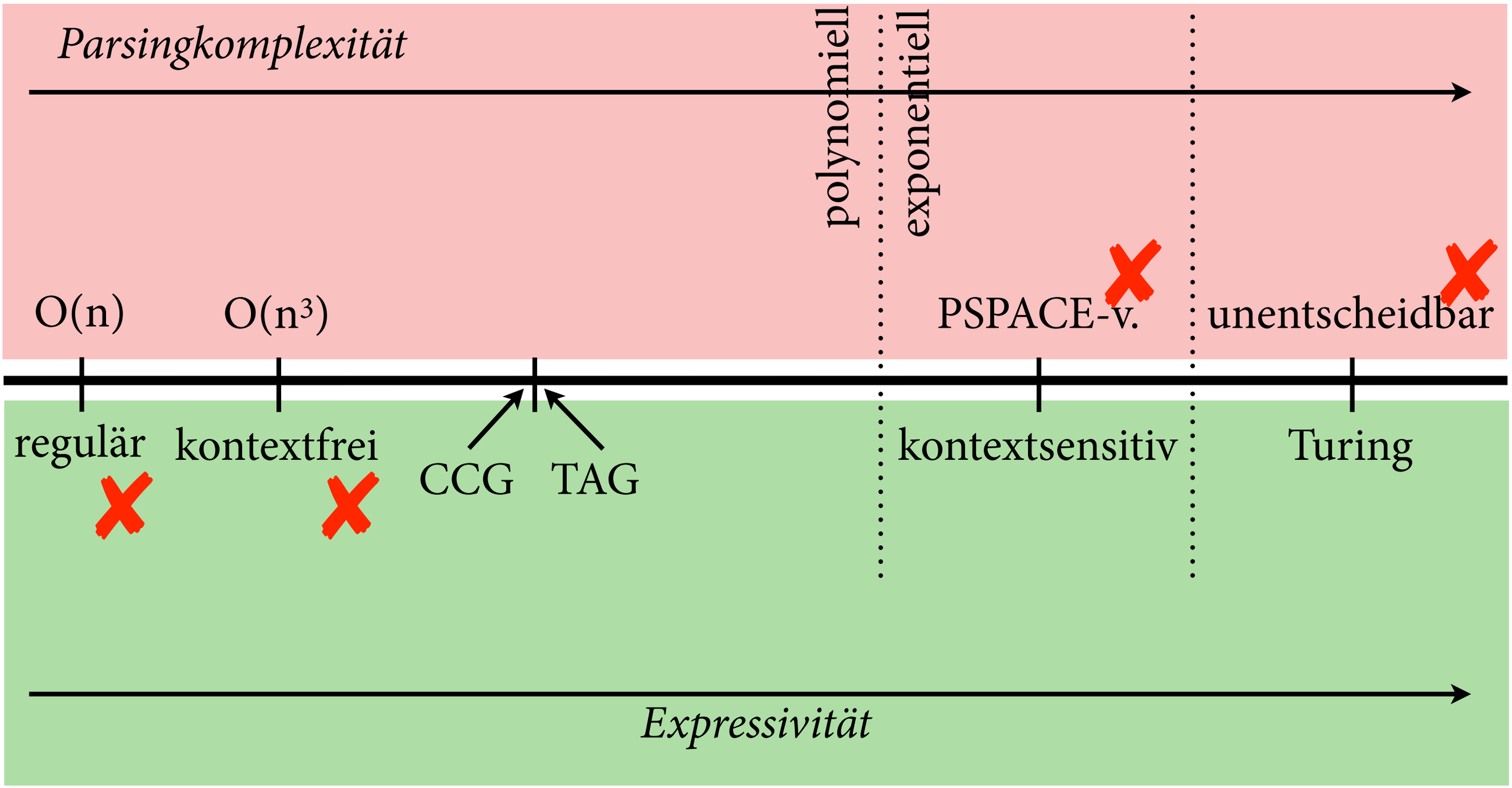
Grammatikformalismen



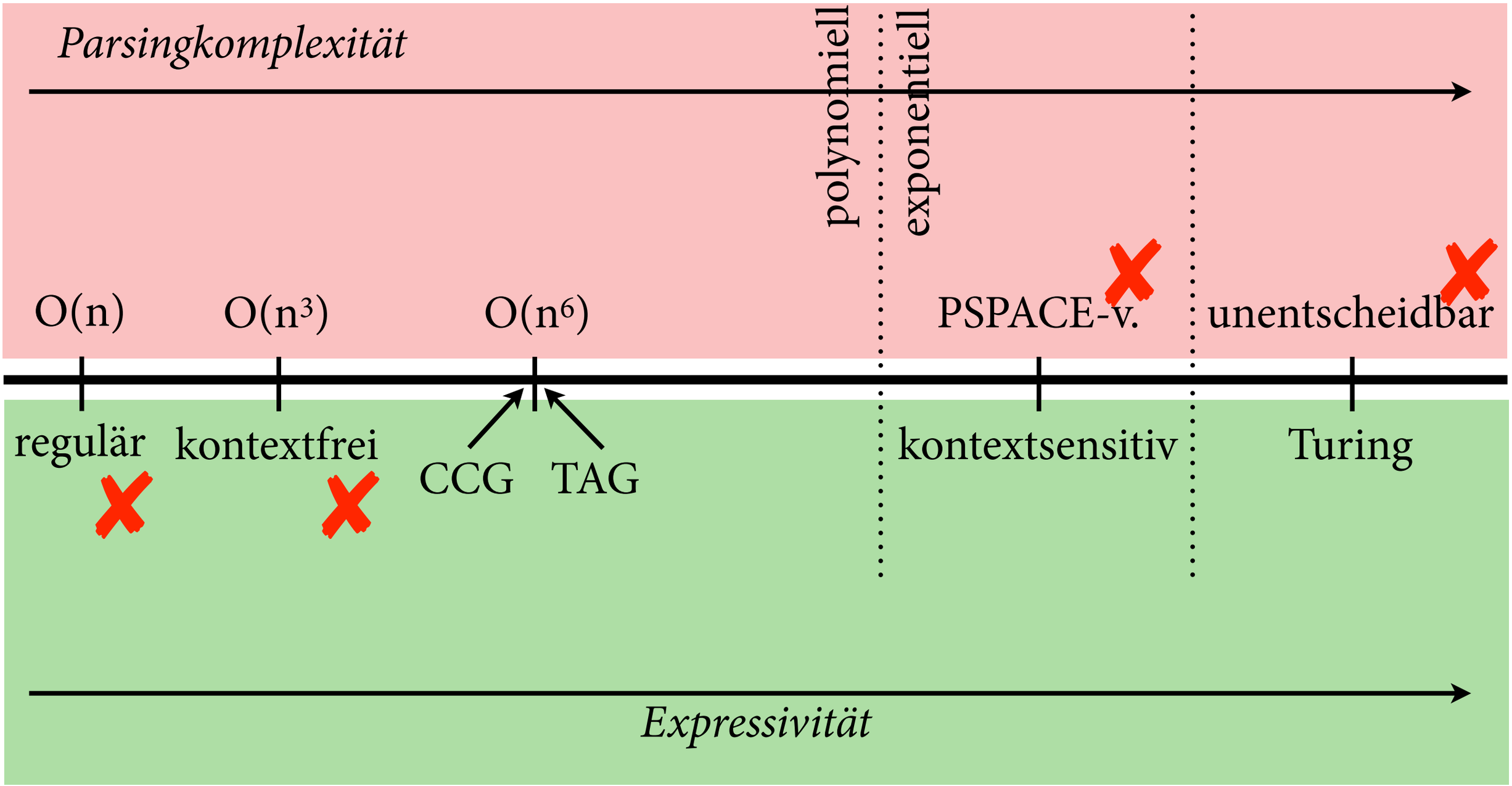
Grammatikformalismen



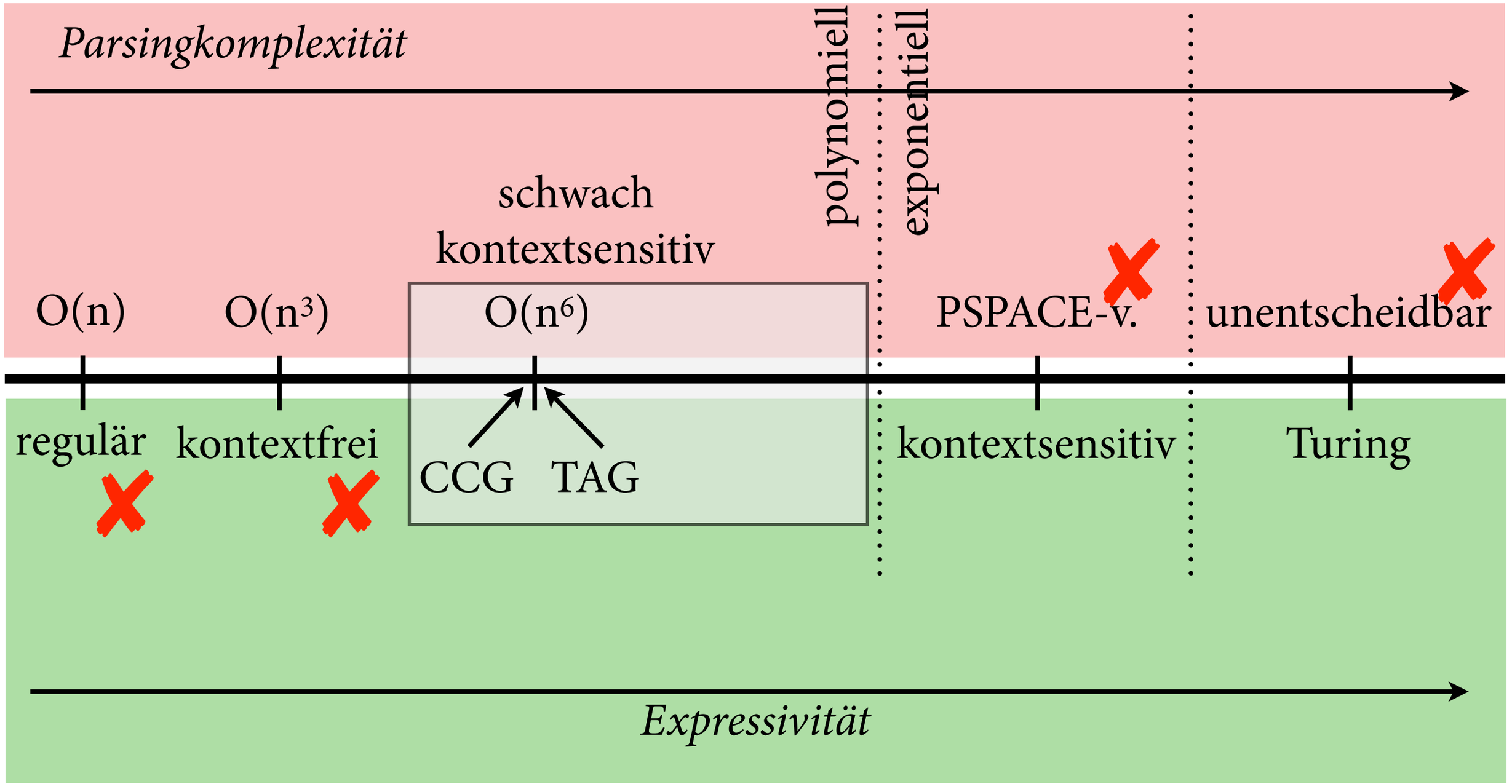
Grammatikformalismen



Grammatikformalismen



Grammatikformalismen



Schwach kontextsensitiv

- Idee von Joshi (1985): Klasse von Gramf., die kfG nur ganz vorsichtig erweitern, aber für natürliche Sprachen ausreichen.

- Ein Gramf. heißt *schwach kontextsensitiv,* wenn
  - er alle kontextfreien Sprachen beschreiben kann
  - man ihn in polynomieller Zeit parsen kann
  - er “eingeschränkte Cross-Serial Dependencies” zulässt
  - er die Constant-Growth-Eigenschaft hat

Cross-Serial Dependencies

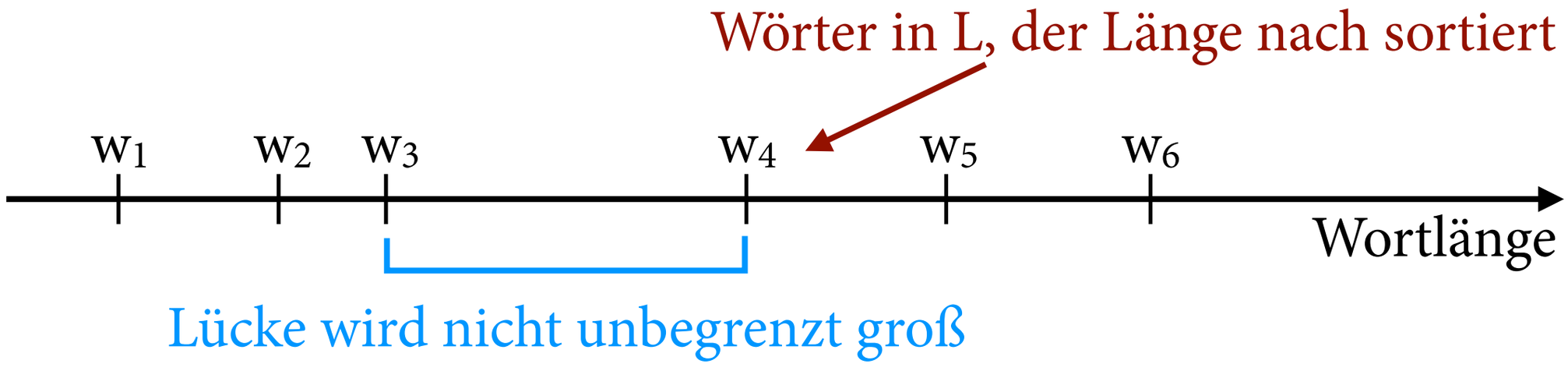
das mer em Hans es huus hälfed aastriiche

- Formale Variante: Grammatikformalismus kann die Copy-Sprache beschreiben.
- Mit kfGen geht das nicht: nur $w^R w$, nicht ww.
- TAG und CCG können zumindest manche CSDs.

Constant Growth

- Intuition: Wenn man alle Wörter einer Sprache der Länge nach ordnet, gibt es keine beliebig großen Lücken.

- Eine Sprache L hat die *constant-growth-Eigenschaft*, wenn es eine Zahl c_0 und eine endliche Menge C von positiven Zahlen gibt, so dass es für jedes Wort $w \in L$ mit $|w| \geq c_0$ ein $c \in C$ sowie ein Wort $w' \in L$ gibt mit $|w'| = |w| + c$.



Constant Growth

- Es haben die Constant-Growth-Eigenschaft:
  - alle kontextfreien Sprachen
  - wohl alle natürlichen Sprachen, weil man jeden Satz ein *kleines* bisschen länger machen kann (Längen von Konstituenten nicht beliebig verteilt)

- Beispiel einer Sprache, die die Eigenschaft *nicht* hat, ist

$$\{a^{2^n} \mid n \geq 1\}.$$

TAG ist schwach k-sensitiv

- TAG (mit Adjunktionsconstraints):
  - enthält alle kf. Sprachen
  - Parsing in $O(n^6)$
  - Cross-Serial-Dependencies ja; COPY geht
  - Constant Growth: ja

- Also ist TAG ein schwach kontextsensitiver Grammatikformalismus.

CCG ist schwach k-sensitiv

- CCG (mit Regeleinschränkungen):
  - enthält alle kf. Sprachen
  - Parsing in $O(n^6)$
  - Cross-Serial-Dependencies ja; COPY geht *
  - Constant Growth: ja

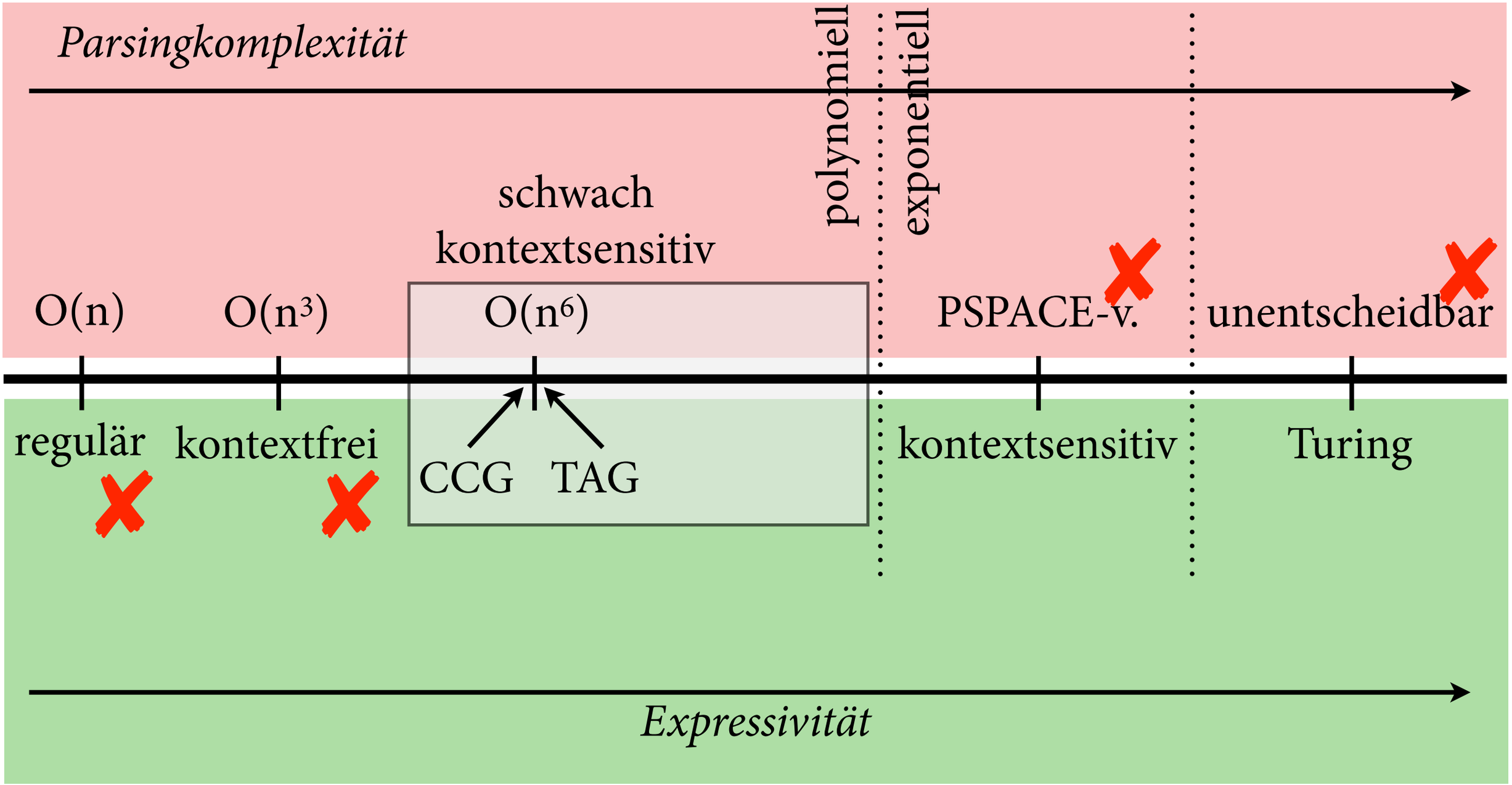
- Also ist CCG ein schwach kontextsensitiver Grammatikformalismus.

*) NB: Geht nur, wenn Regeleinschränkungen erlaubt sind!

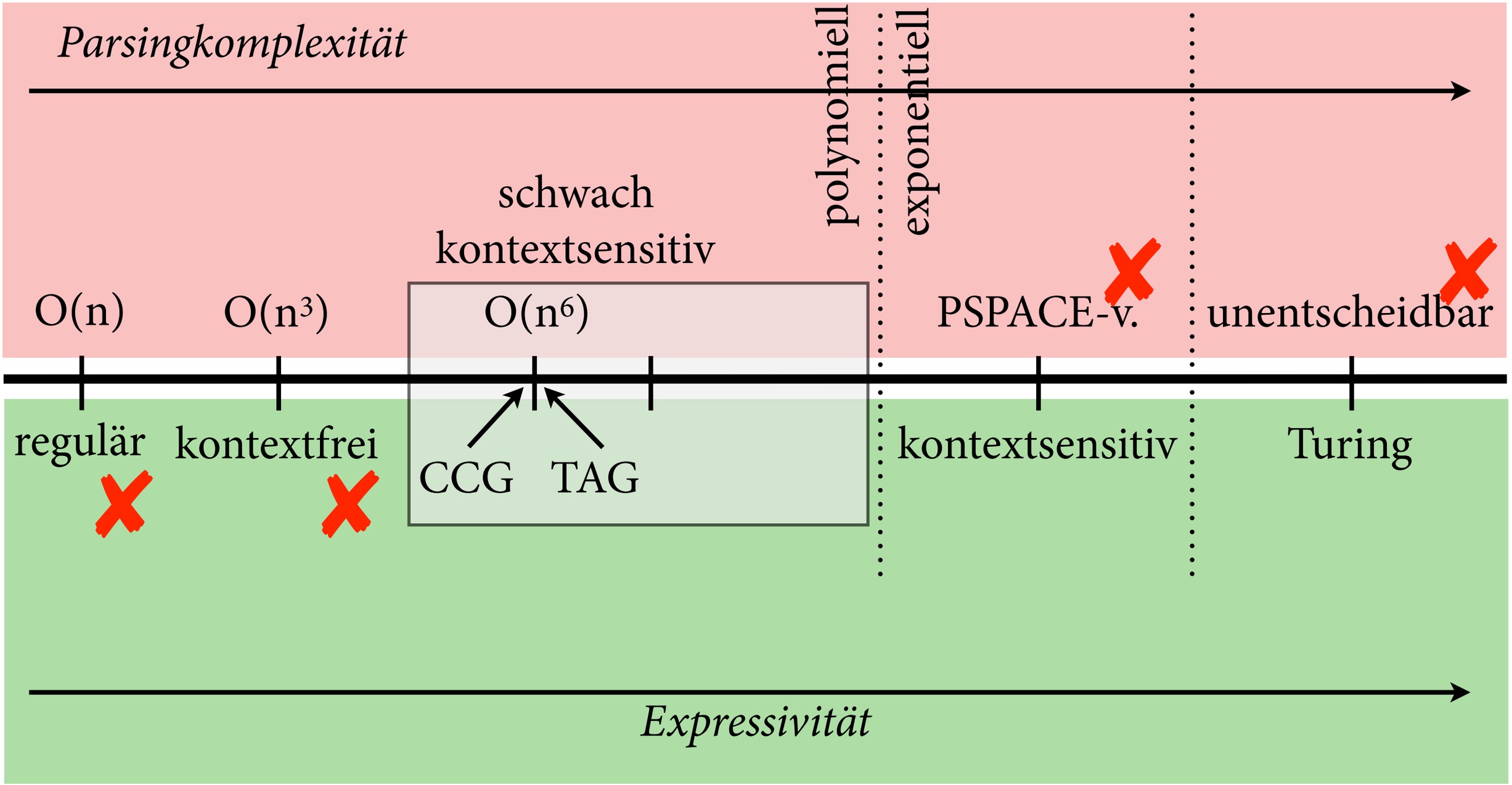
Größere sks. Formalismen

- TAG und CCG sind nicht die expressivsten schwach kontext-sensitiven Formalismen.
  - Beispiel: ein (hypothetischer) Formalismus, der alle TAG-Grammatiken erlaubt plus eine Grammatik für COUNT(5).
- Wichtige expressivere Formalismen:
  - Linear Context-Free Rewrite Systems (LCFRS; Weir)
  - Range Concatenation Grammars (RCG; Boullier)
  - Reguläre Dependenzgrammatiken (RDG; Kuhlmann)

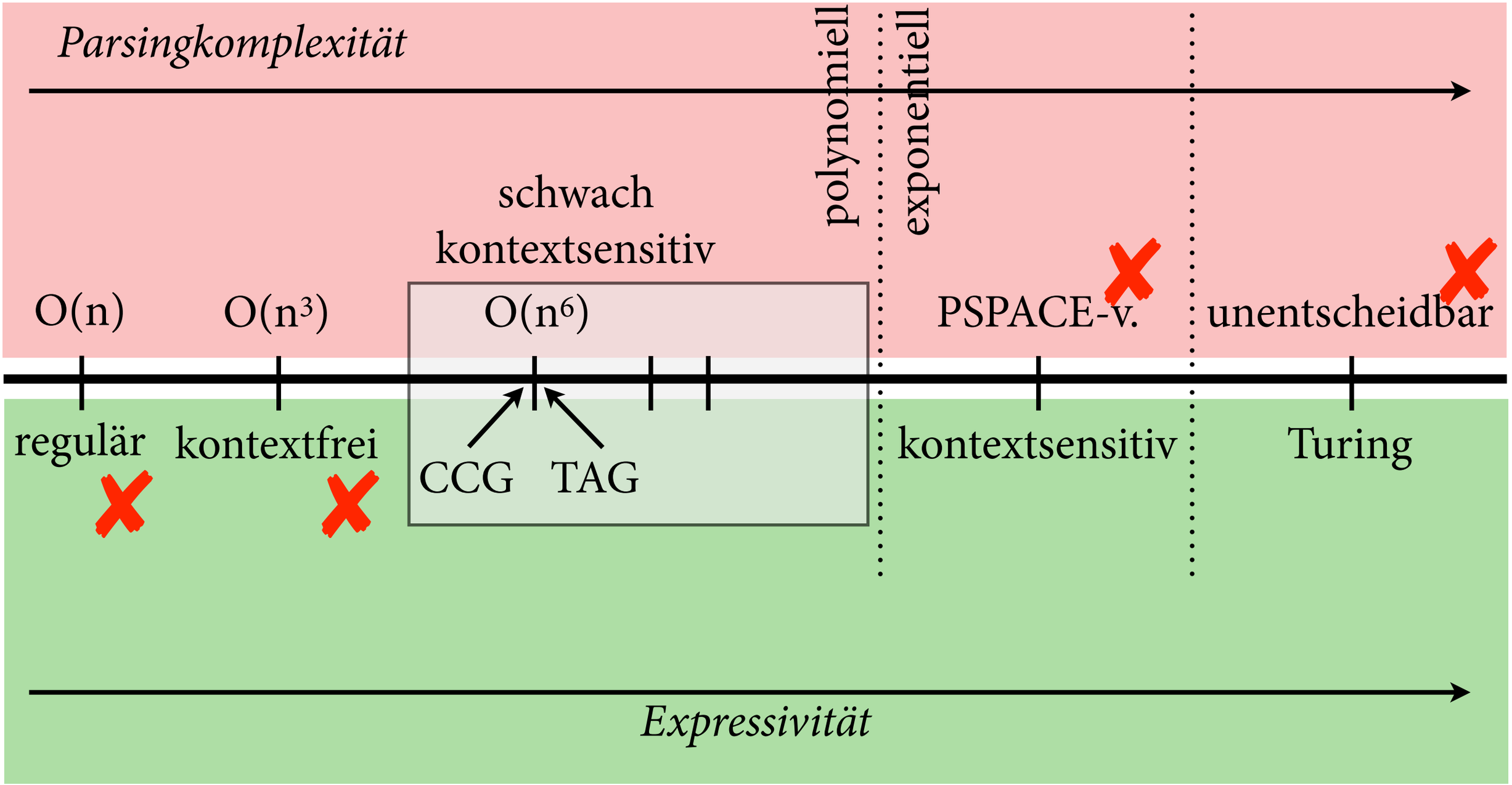
Eine allgemeinere Perspektive



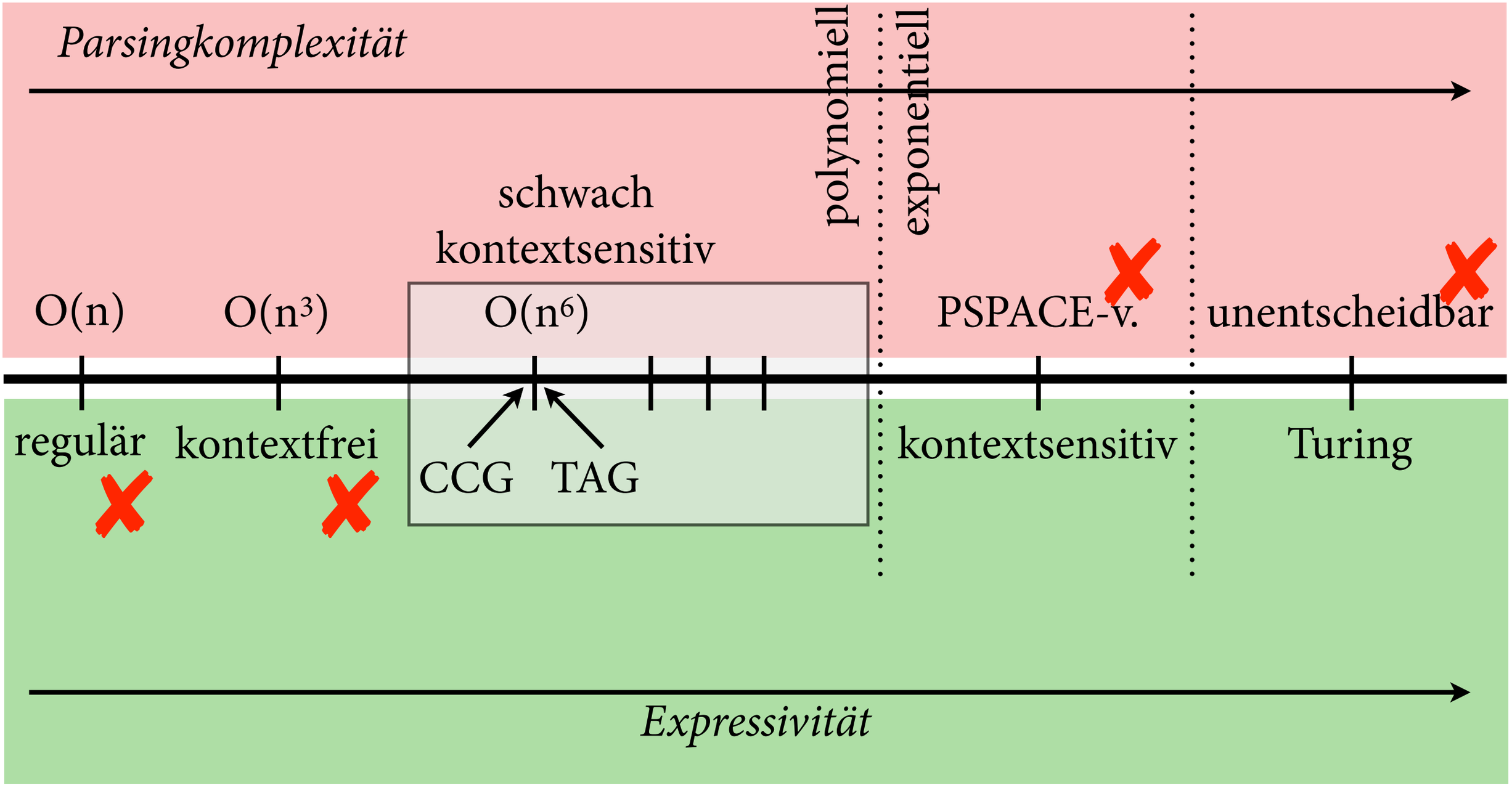
Eine allgemeinere Perspektive



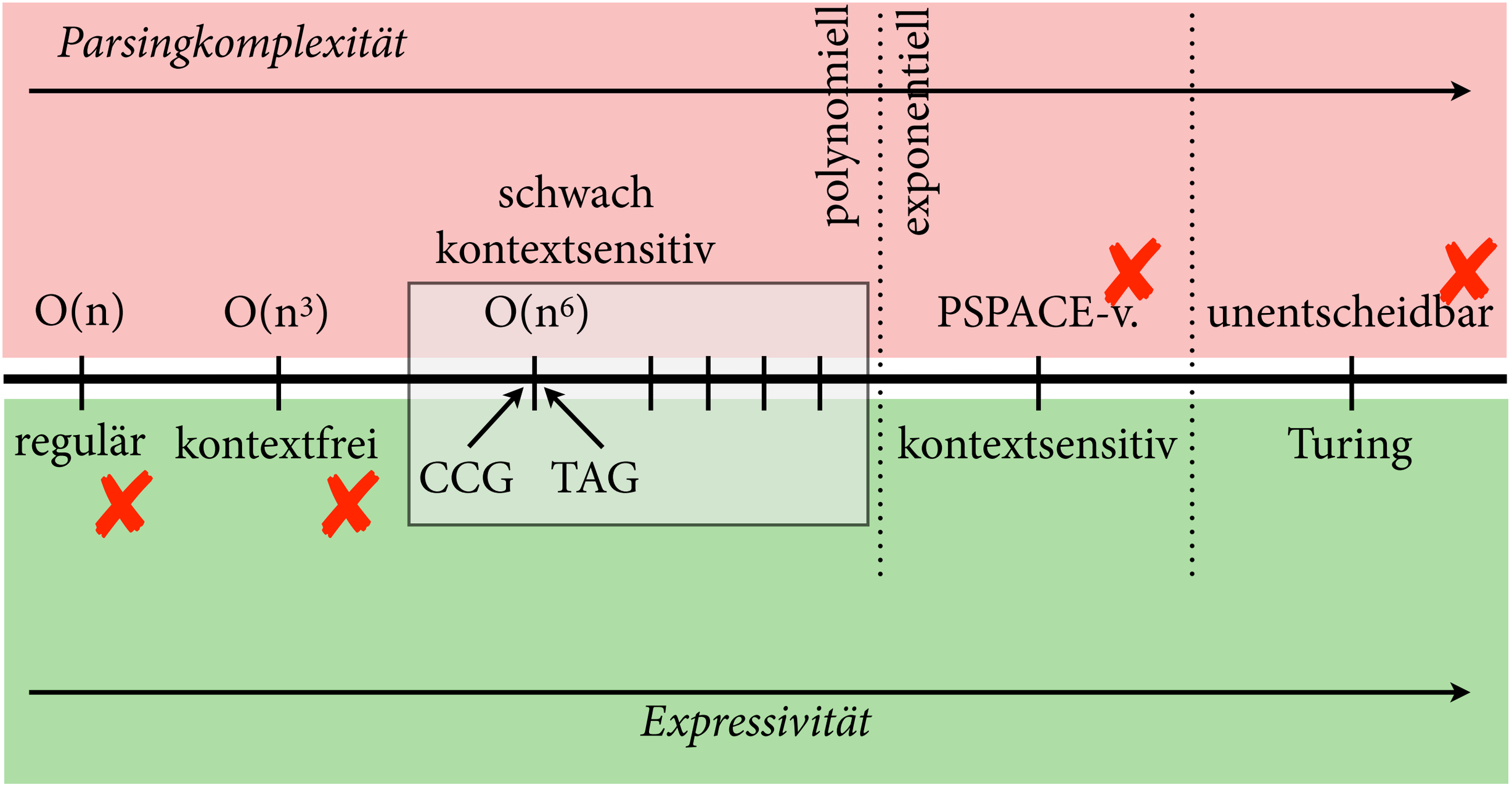
Eine allgemeinere Perspektive



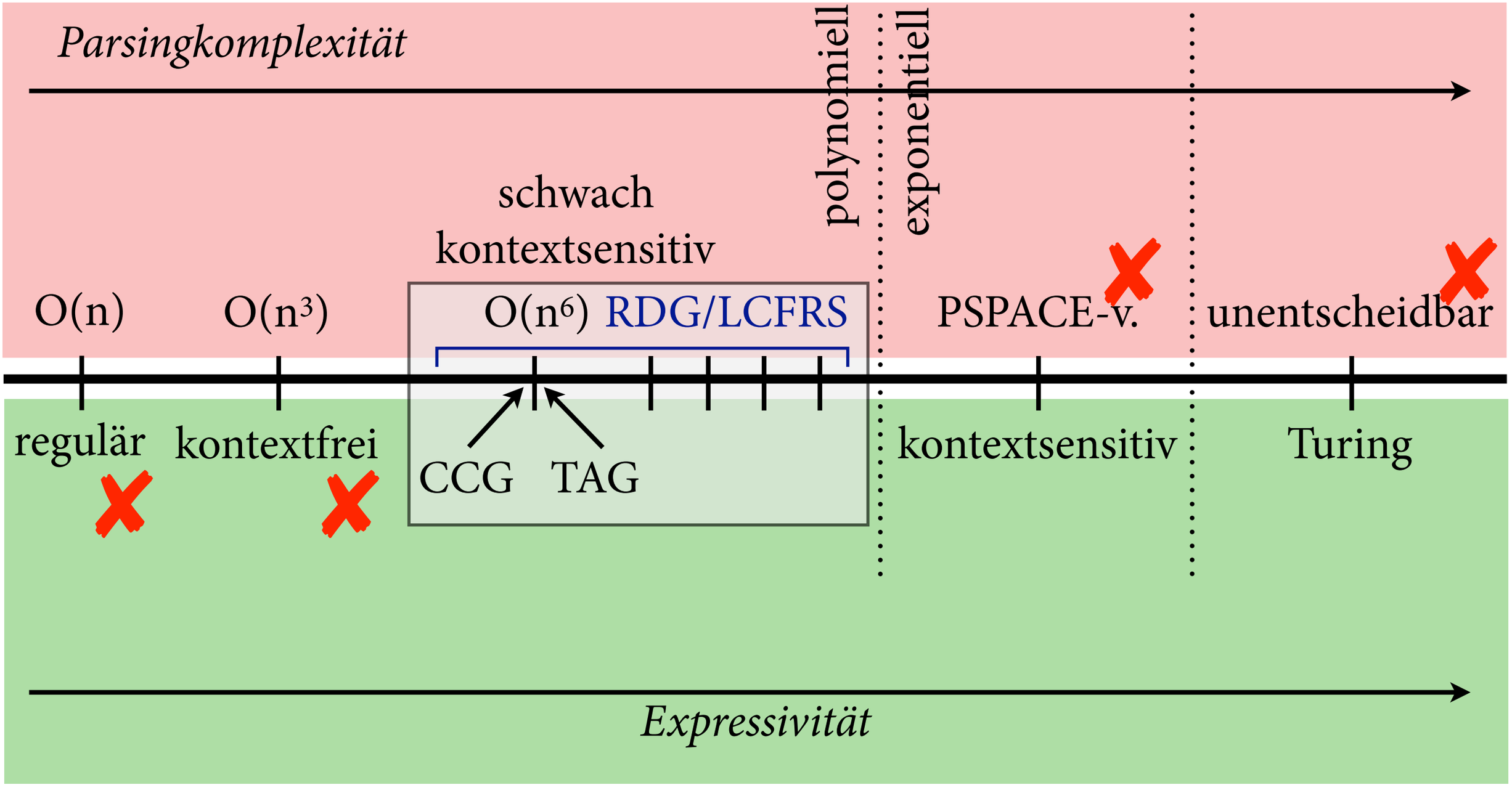
Eine allgemeinere Perspektive



Eine allgemeinere Perspektive



Eine allgemeinere Perspektive



Reguläre Baumsprachen

- Nächstes Mal: reguläre Baumsprachen (RTLs):
  - Sprachen von Bäumen
  - kleine Sprachklasse; analog zu regulären Stringsprachen
- RTLs sind Grundlage von LCFRS, RDG, IRTG.

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.

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

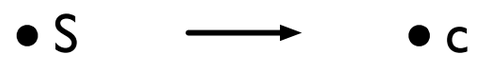
RTG G

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

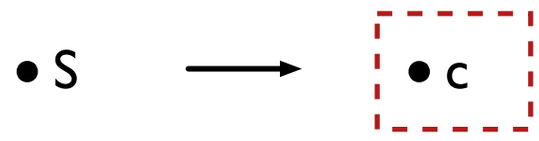


Ableitungen von RTGs

- 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

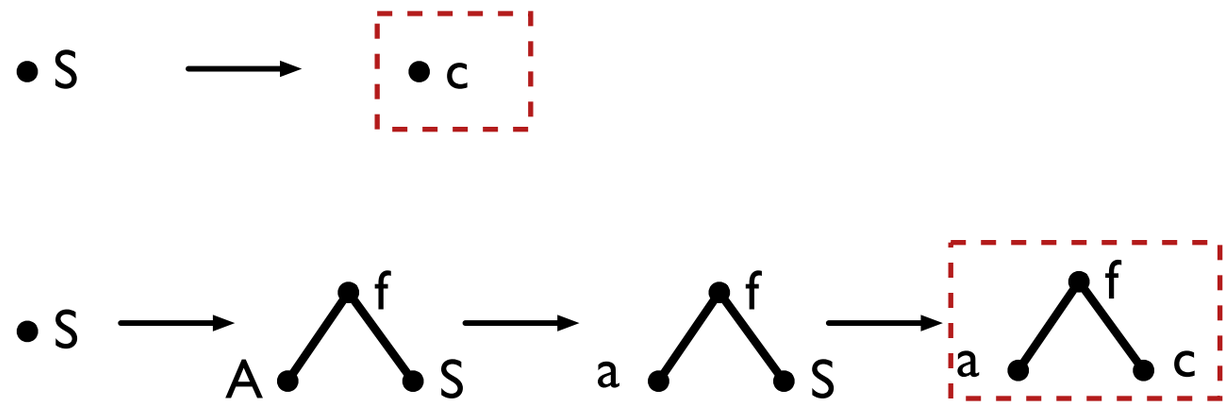


Ableitungen von RTGs

- 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

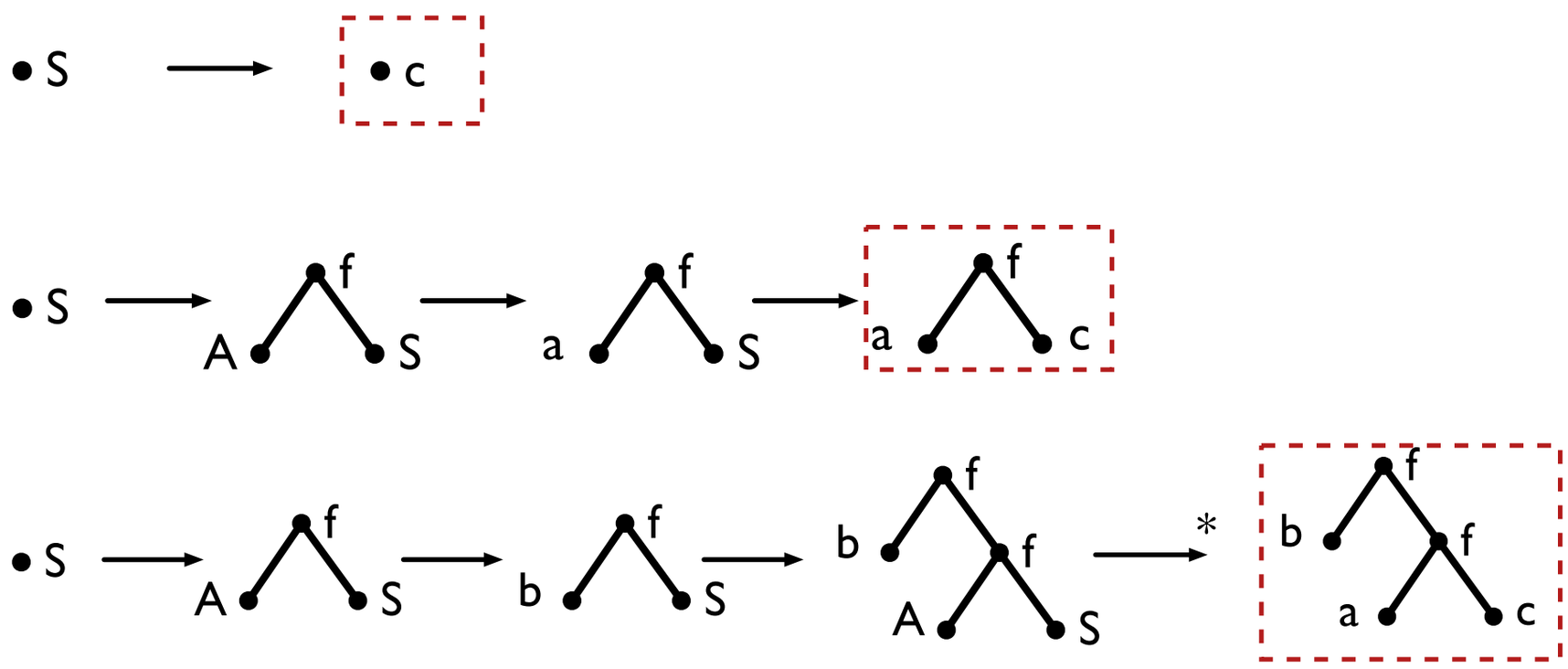


Ableitungen von RTGs

- 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



Reguläre Baumsprachen

- Eine Sprache L von Bäumen heißt *regulär,* wenn es eine RTG G gibt mit L = L(G).

- Man kann reguläre Baumsprachen (RTLs) auch über *Baumautomaten* definieren.

- RTLs sind abgeschlossen unter Vereinigung, Schnitt, Komplement, usw.

- Sprache der Parsebäume einer kfG ist immer regulär. Sprache der Erträge einer RTL ist immer kontextfrei.

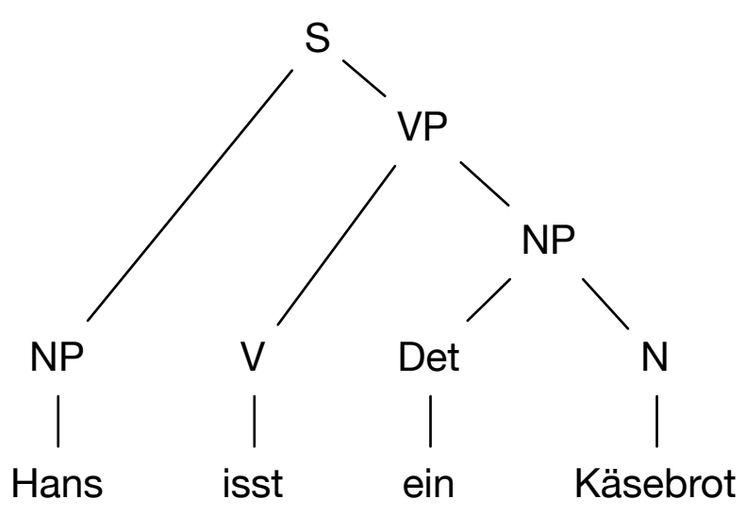
kfG ↔ RTG

S → NP VP	V → isst
NP → Det N	NP → Hans
VP → V NP	Det → ein
	N → Käsebrot

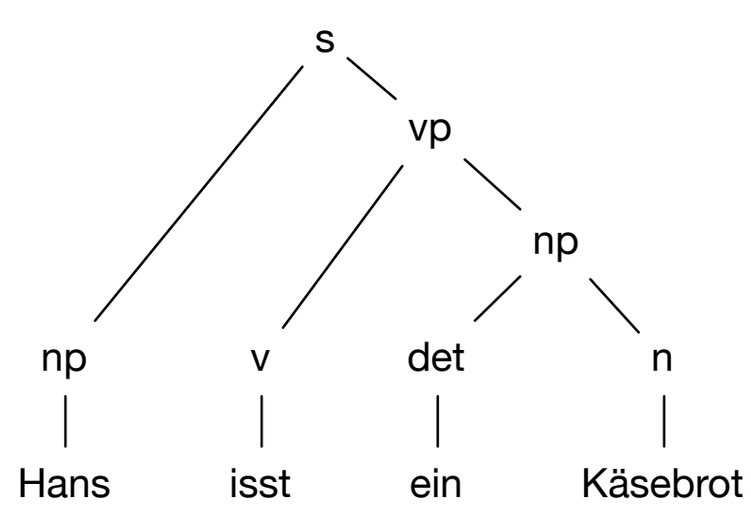
kontextfreie Grammatik G_1

S → s(NP, VP)	V → v(isst)
NP → np(Det, N)	NP → np(Hans)
VP → vp(V, NP)	Det → det(ein)
	N → n(Käsebrot)

RTG G_2



“Hans isst ein Käsebrot”
$\in L(G_1)$
(Sprache enthält Strings)



$\in L(G_2)$
(Sprache enthält Bäume)

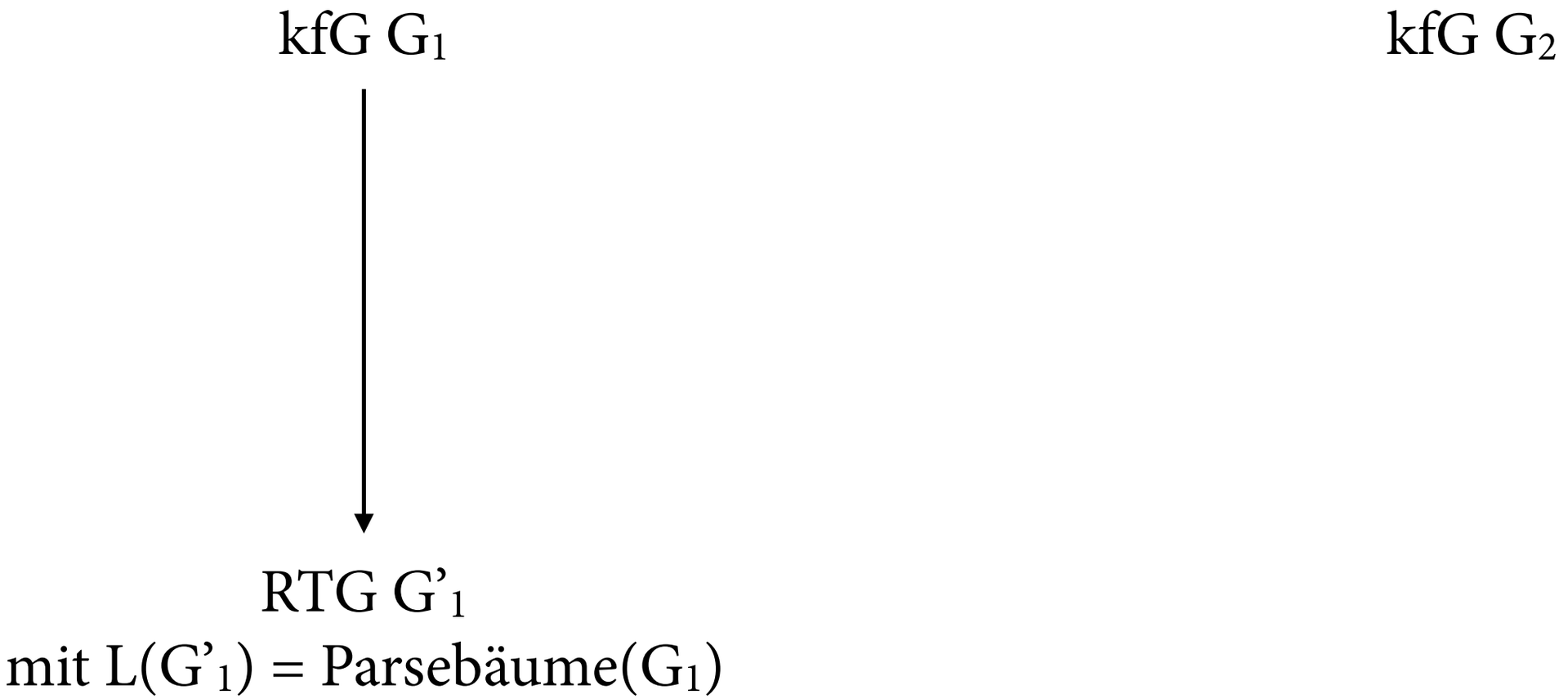
Ein Rätsel

Fakt: Kontextfreie Sprachen sind
nicht unter Schnitt abgeschlossen.

kfG G_1 kfG G_2

Ein Rätsel

Fakt: Kontextfreie Sprachen sind
nicht unter Schnitt abgeschlossen.

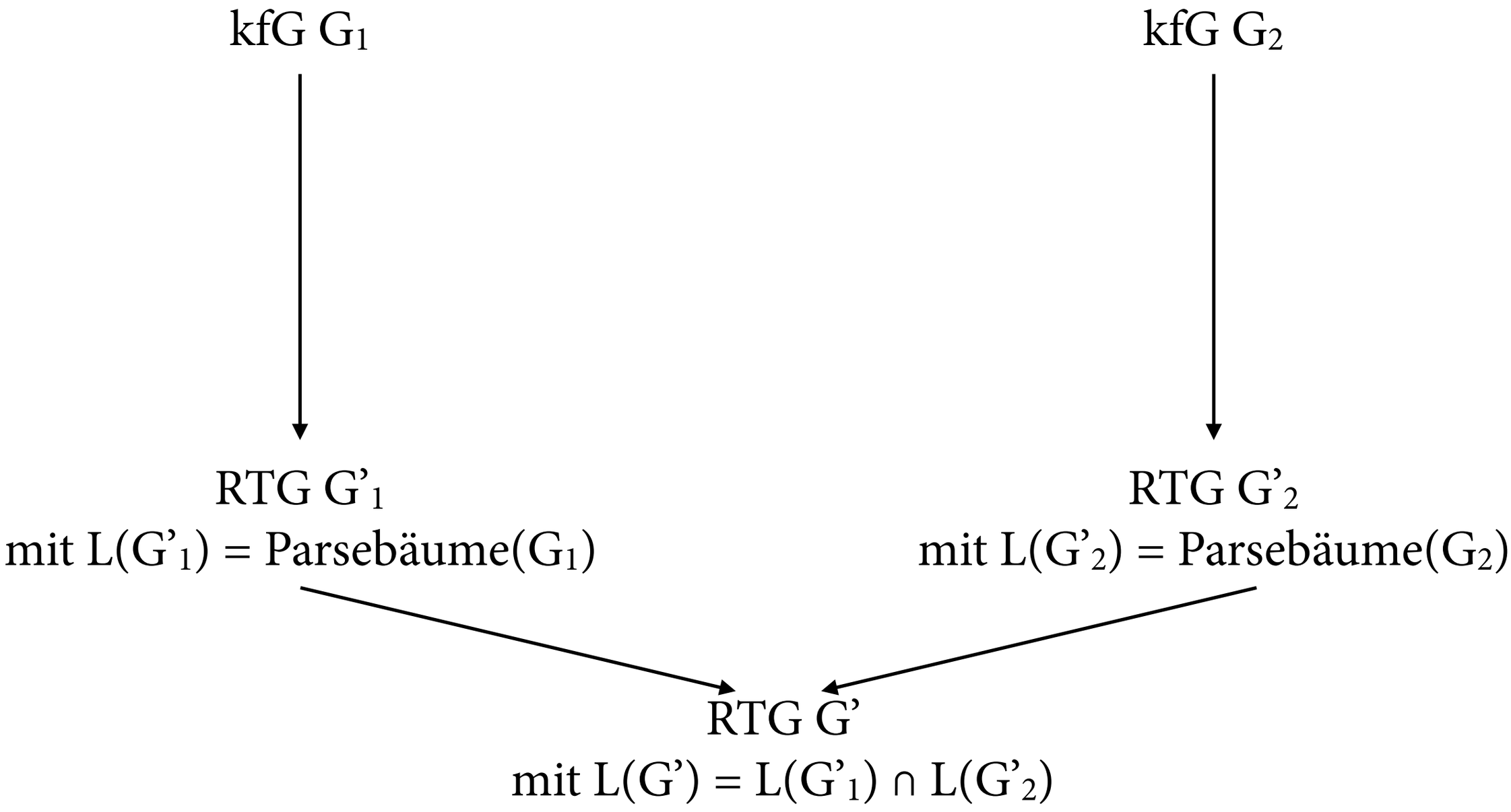


Ein Rätsel

Fakt: Kontextfreie Sprachen sind
nicht unter Schnitt abgeschlossen.

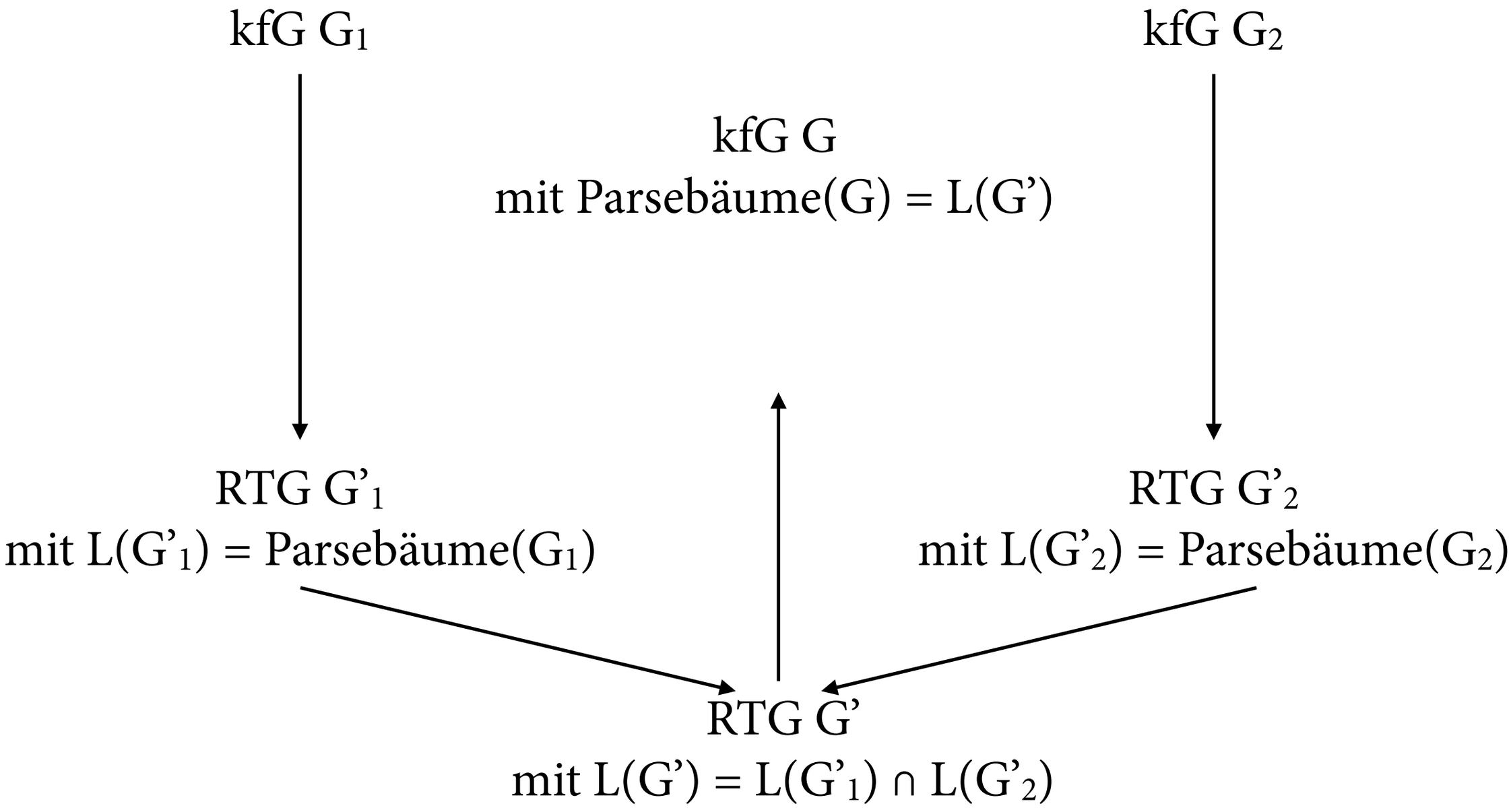
Ein Rätsel

Fakt: Kontextfreie Sprachen sind
nicht unter Schnitt abgeschlossen.



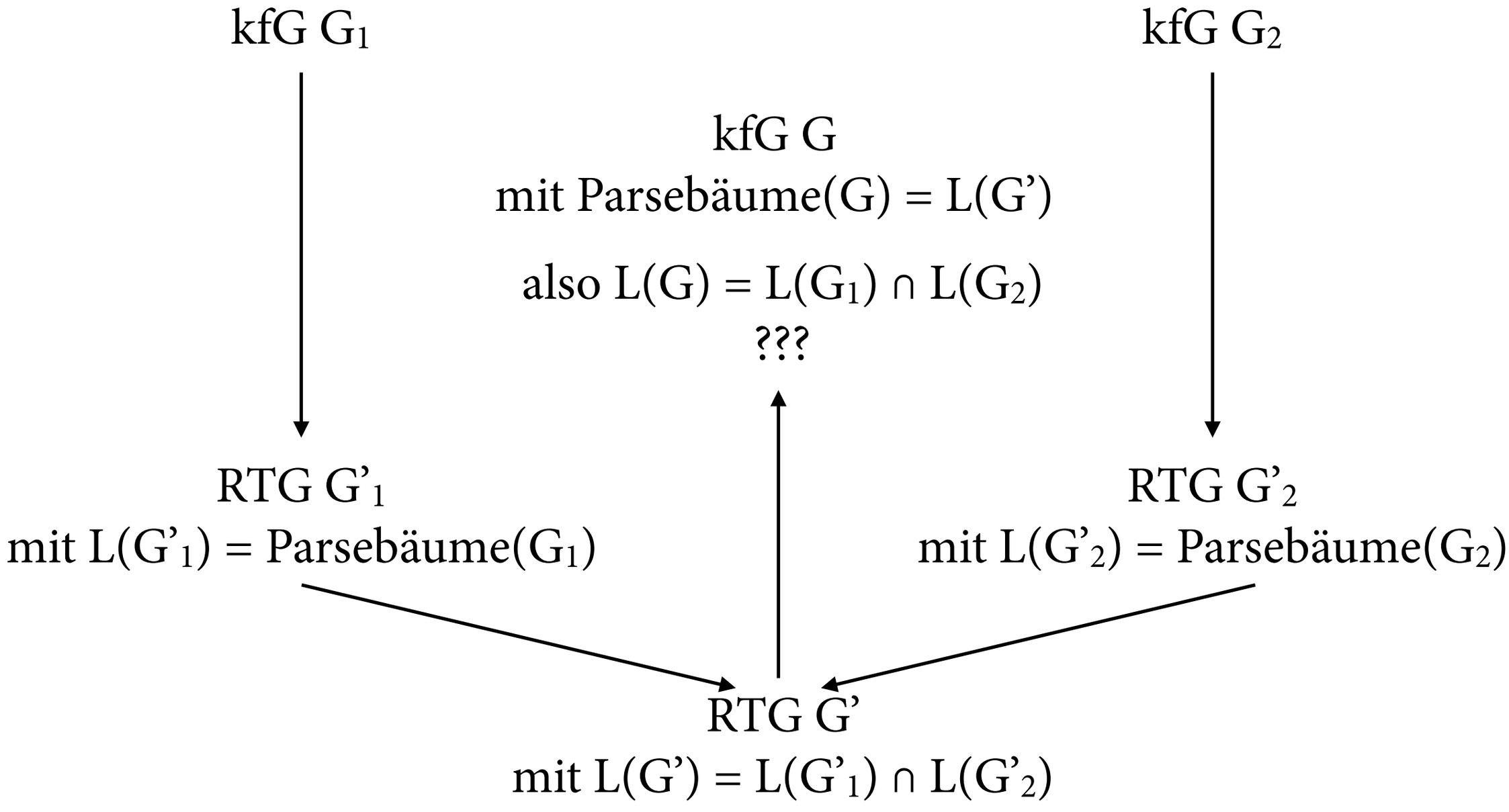
Ein Rätsel

Fakt: Kontextfreie Sprachen sind
nicht unter Schnitt abgeschlossen.



Ein Rätsel

Fakt: Kontextfreie Sprachen sind
nicht unter Schnitt abgeschlossen.



Zusammenfassung

- Äquivalenz von TAG und CCG.
- Schwach kontextsensitive Grammatikformalismen.
- Reguläre Baumsprachen.