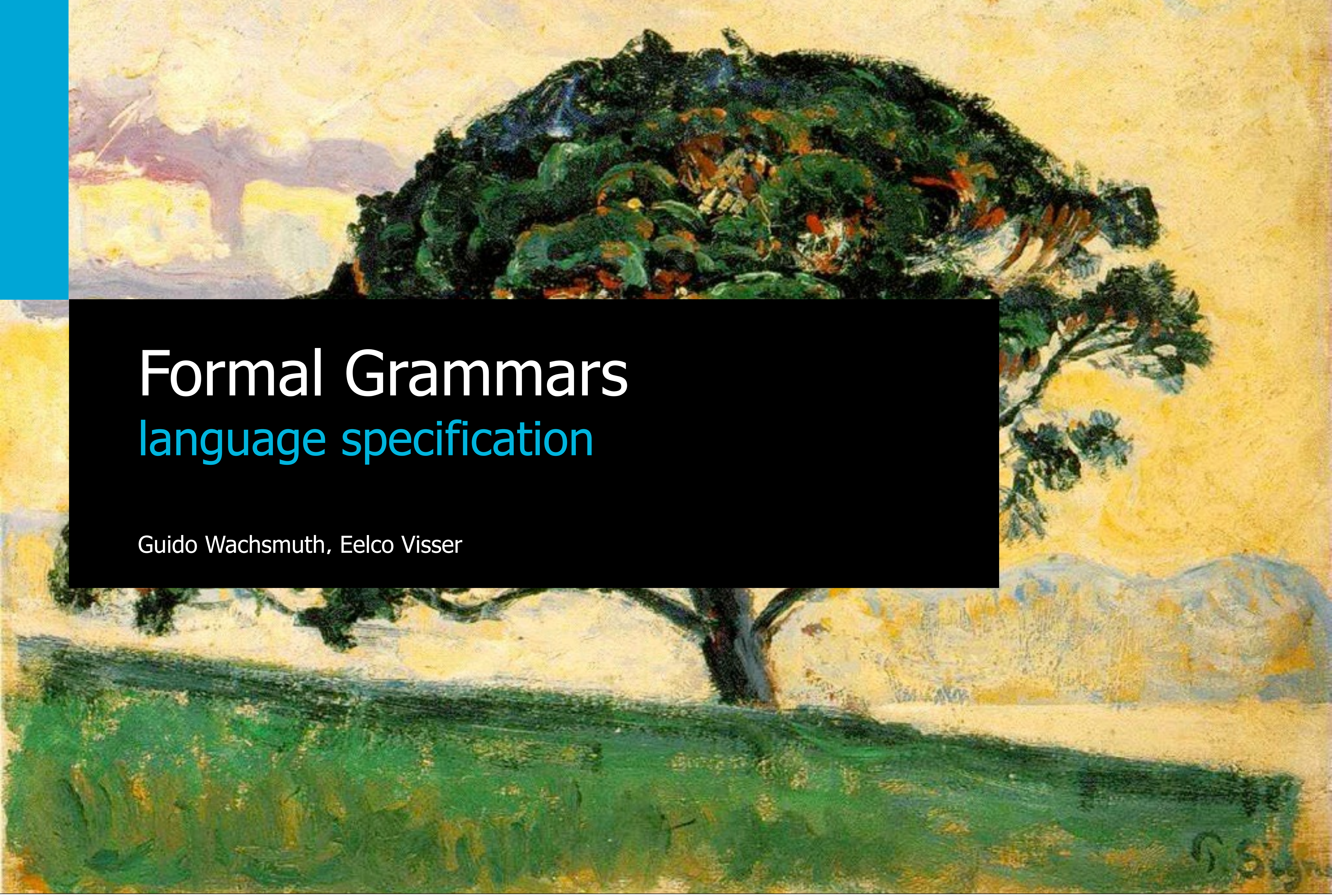


Formal Grammars

language specification

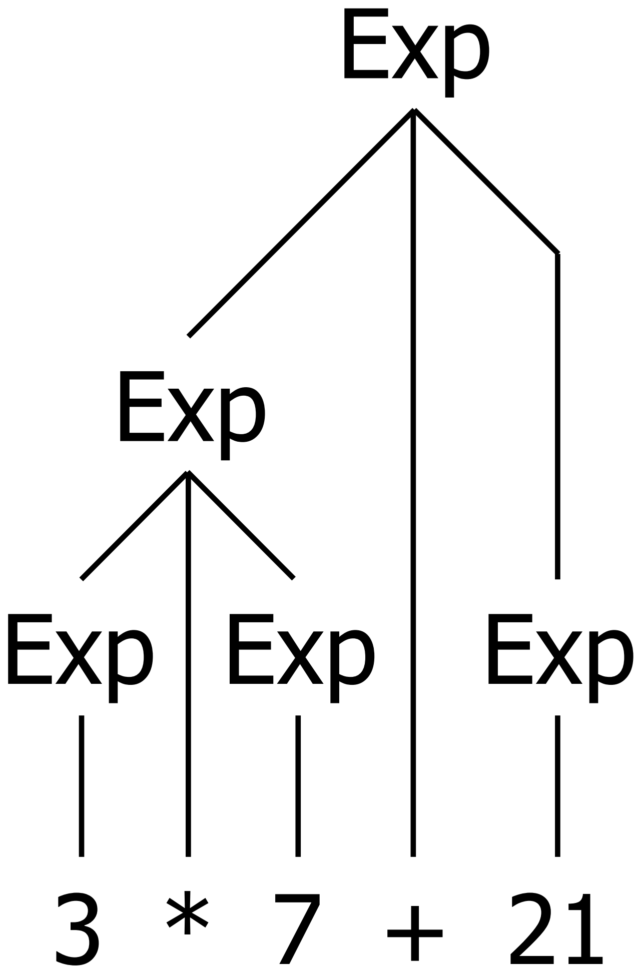
Guido Wachsmuth, Eelco Visser

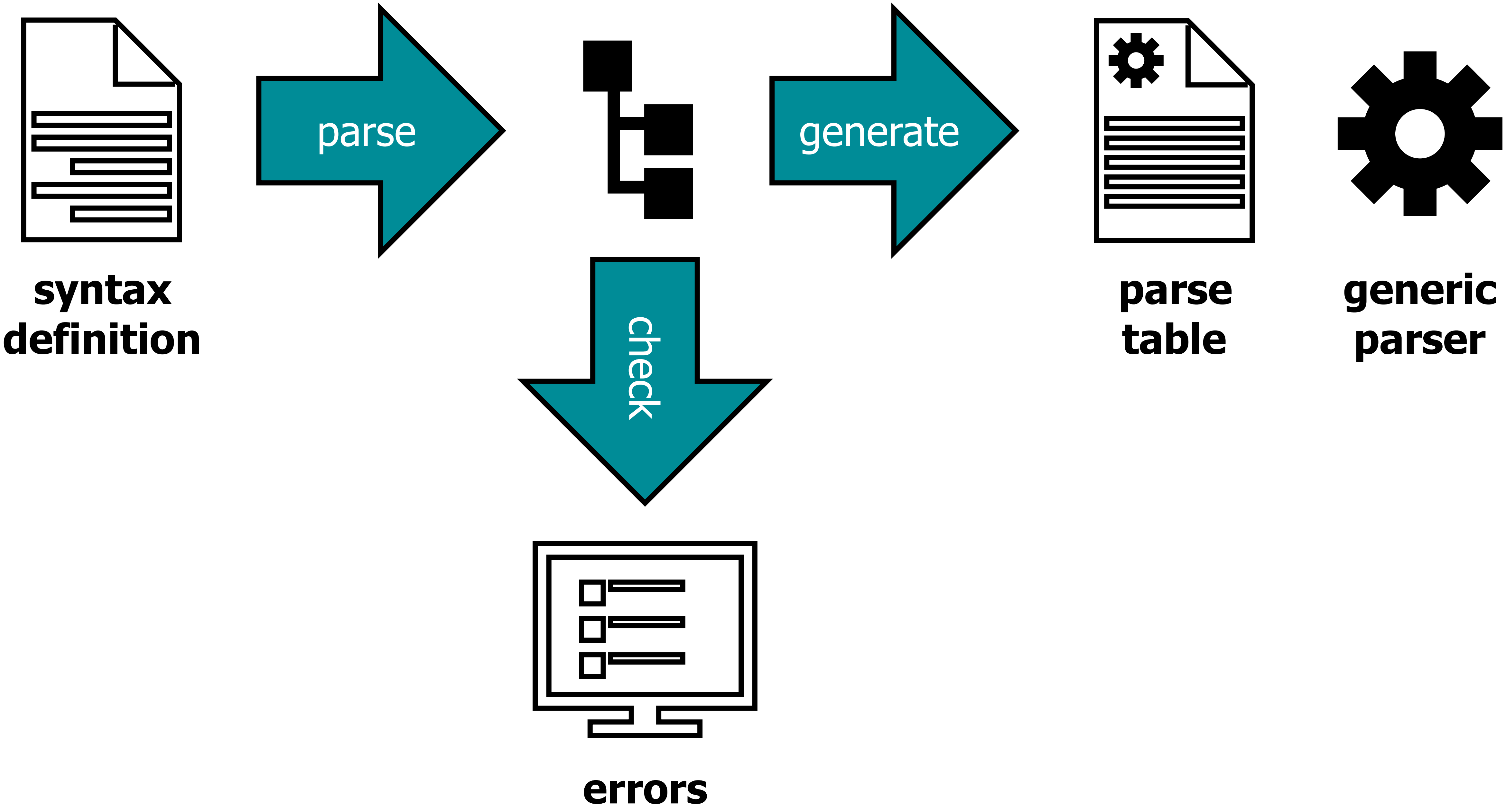
TU Delft

IN4303 2016-2017
Compiler Construction

3 * 7 + 21

parser





syntax definition

parse

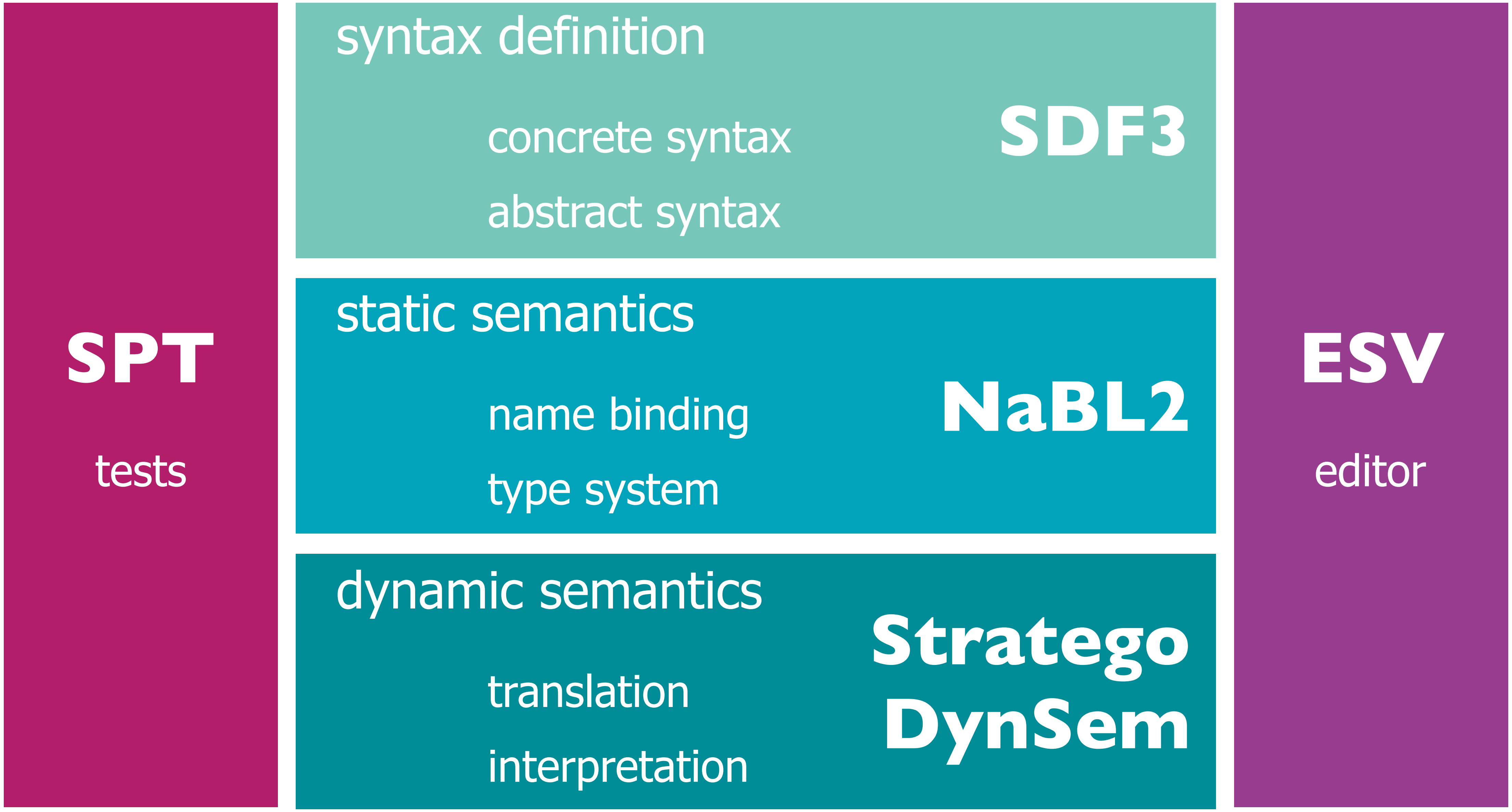
generate

parse table

generic parser

check

errors



SPT

tests

syntax definition

concrete syntax

abstract syntax

SDF3

static semantics

name binding

type system

NaBL2

dynamic semantics

translation

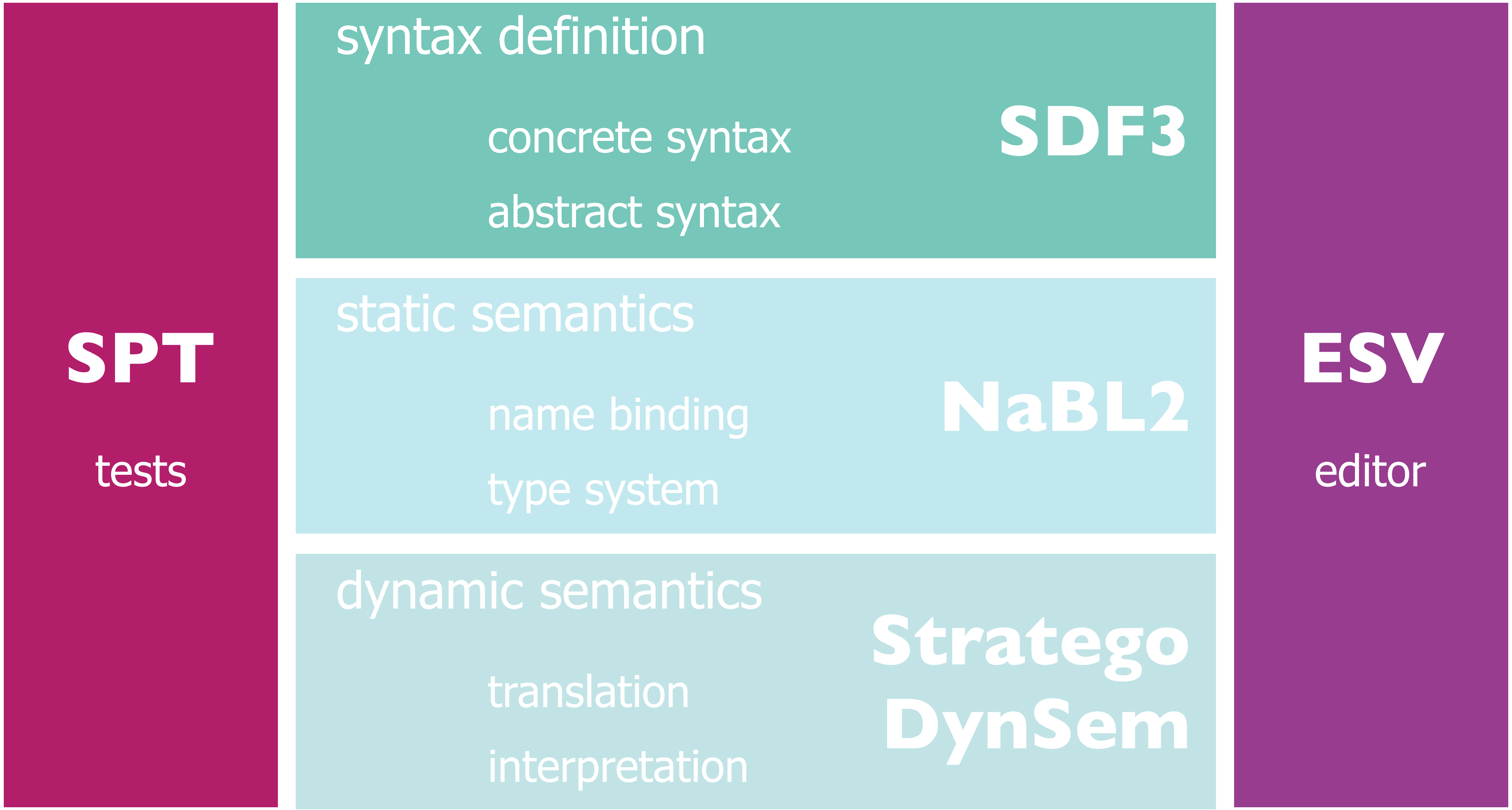
interpretation

Stratego

DynSem

ESV

editor



SPT

tests

syntax definition

concrete syntax

abstract syntax

SDF3

static semantics

name binding

type system

NaBL2

dynamic semantics

translation

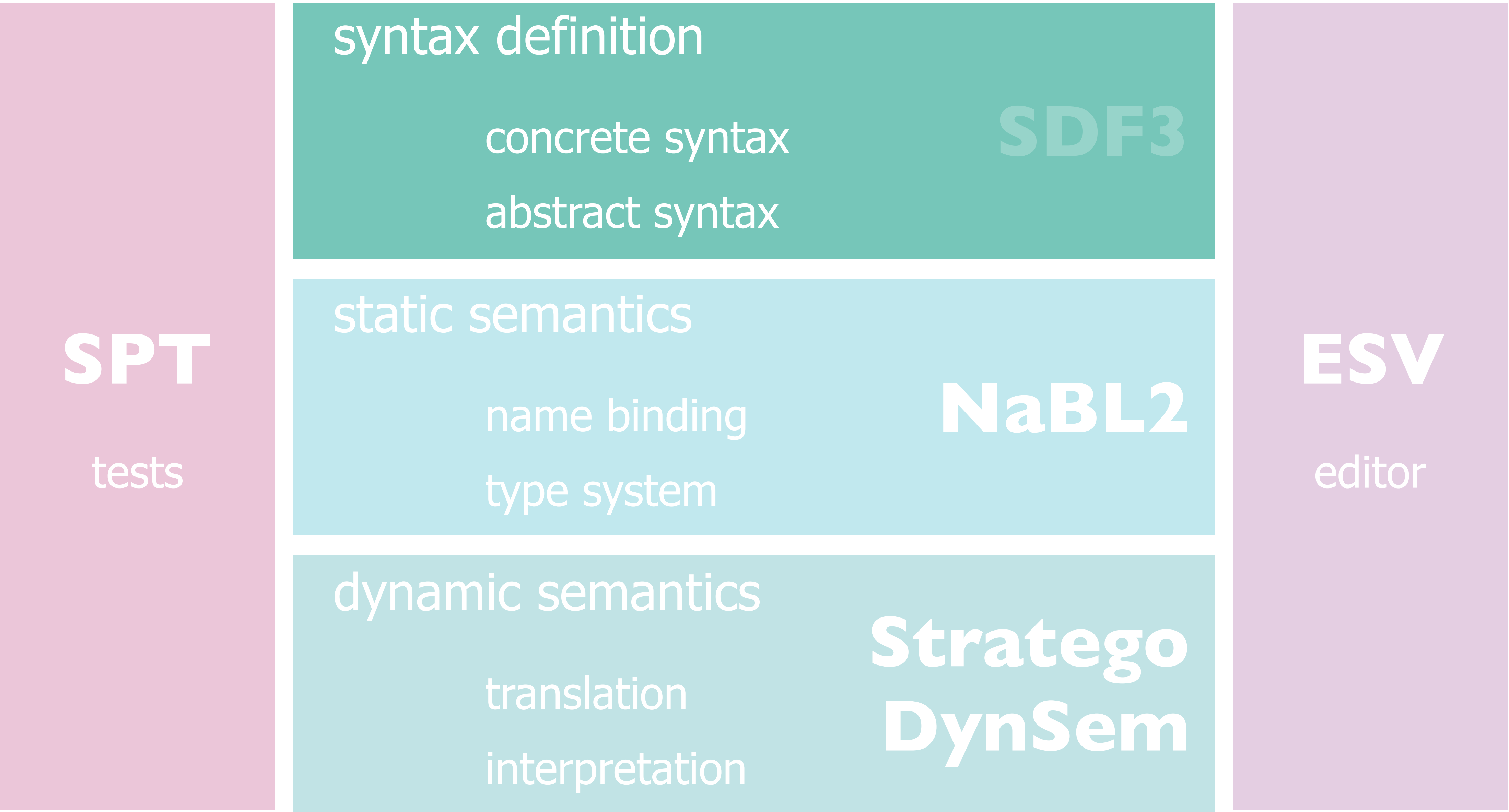
interpretation

Stratego

DynSem

ESV

editor



SPT

tests

syntax definition

concrete syntax

abstract syntax

SDF3

static semantics

name binding

type system

NaBL2

dynamic semantics

translation

interpretation

Stratego

DynSem

ESV

editor

PARENTAL

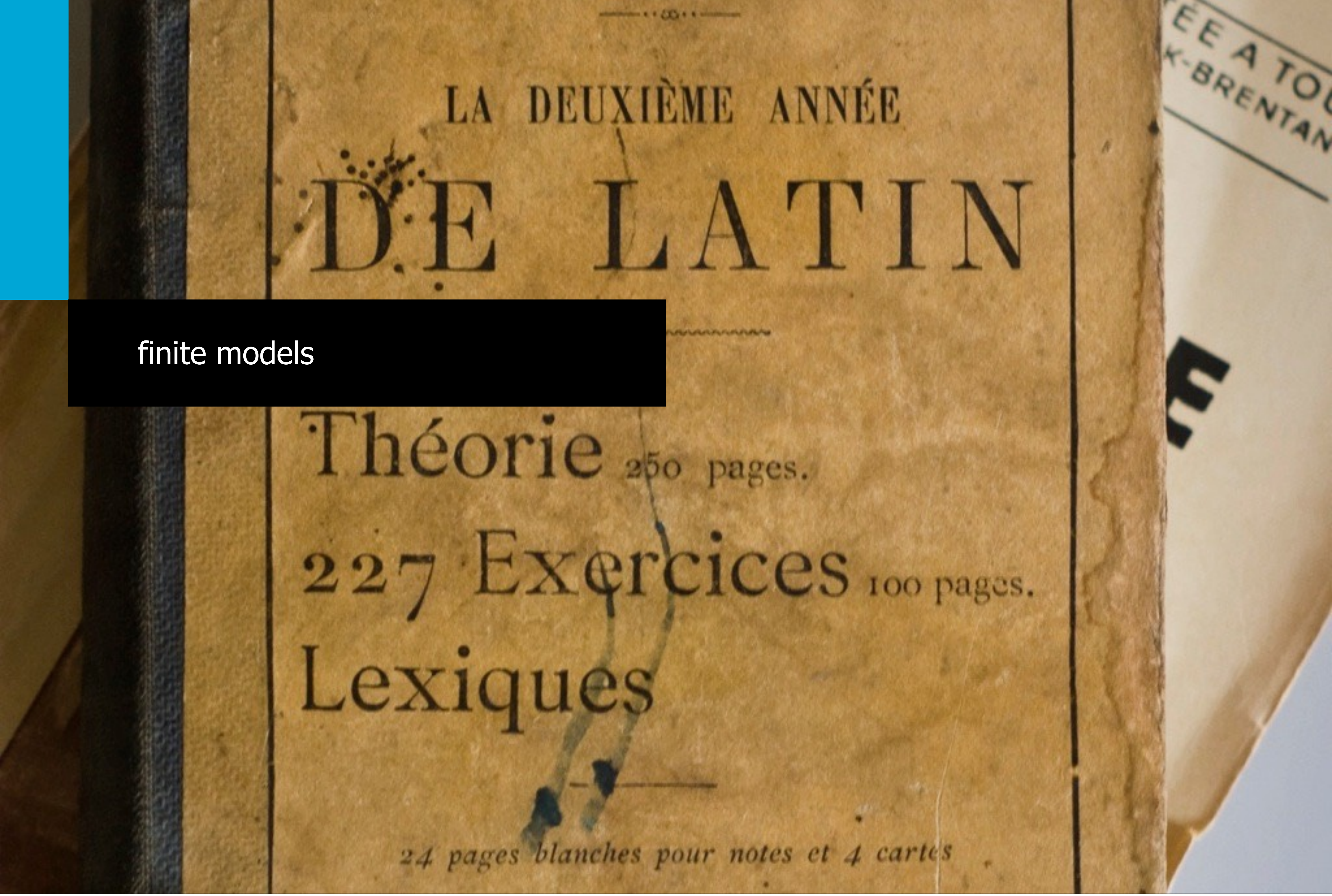
ADVISORY

THEORETICAL CONTENT

formal languages

infinite productivity

finite models



LA DEUXIÈME ANNÉE

DE LATIN

Théorie 250 pages.

227 Exercices 100 pages.

Lexiques

24 pages blanches pour notes et 4 cartes

Philosophy

Linguistics

lexicology

grammar

morphology

syntax

phonology

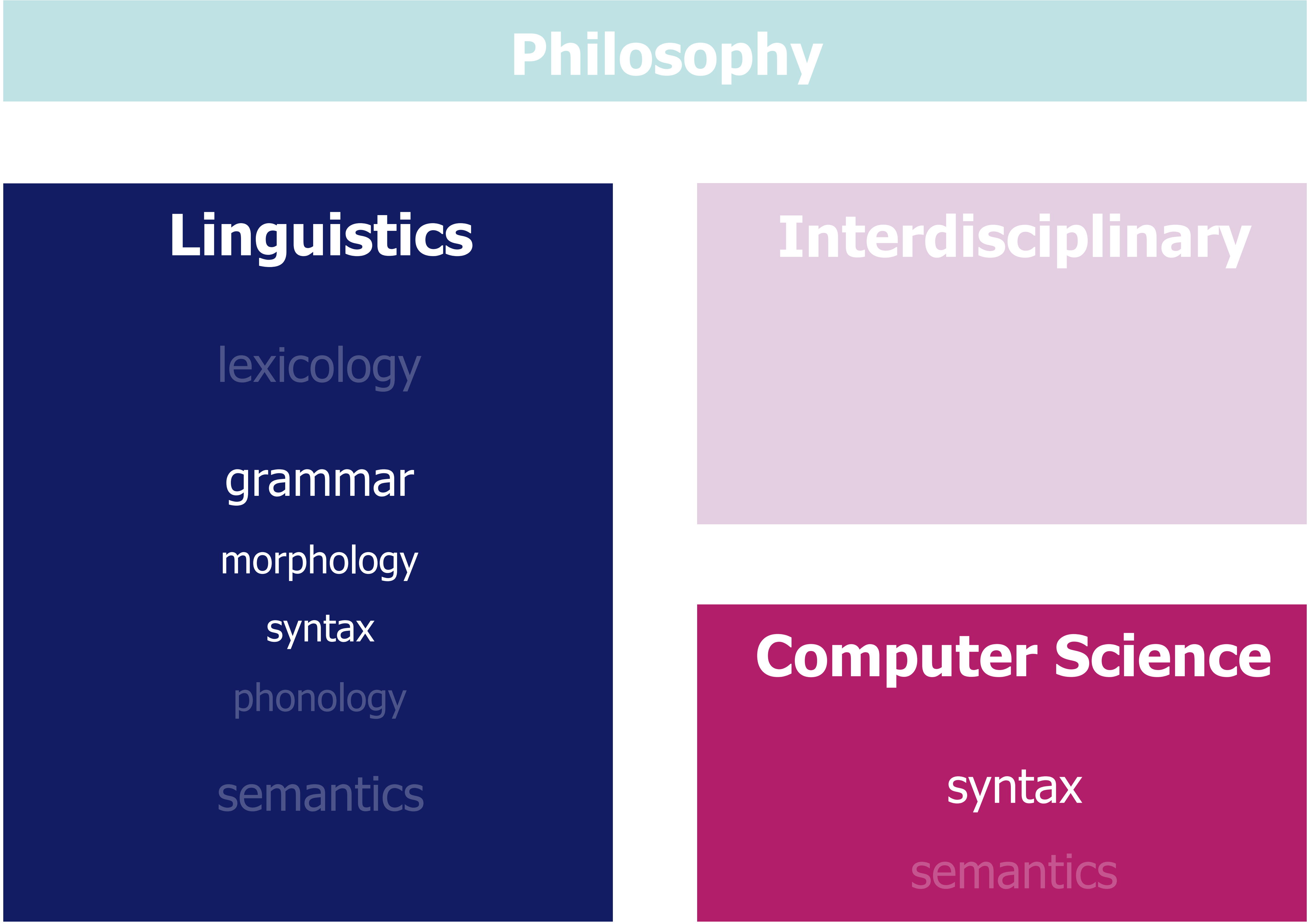
semantics

Interdisciplinary

Computer Science

syntax

semantics



Philosophy

Linguistics

lexicology

grammar

morphology

syntax

phonology

semantics

Interdisciplinary

Computer Science

syntax

semantics

vocabulary Σ

finite, nonempty set of elements (words, letters)

alphabet

vocabulary Σ

- finite, nonempty set of elements (words, letters)
- alphabet

string over Σ

- finite sequence of elements chosen from Σ
- word, sentence, utterance

vocabulary Σ

finite, nonempty set of elements (words, letters)

alphabet

string over Σ

finite sequence of elements chosen from Σ

word, sentence, utterance

formal language λ

set of strings over a vocabulary Σ

$\lambda \subseteq \Sigma^*$

formal grammars

formal grammar G

derivation relation $\Rightarrow_G$

formal language $L(G) \subseteq \Sigma^*$

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$$

decimal numbers

morphology

G = (N, Σ, P, S)

Num → Digit Num
Num → Digit
Digit → “0”
Digit → “1”
Digit → “2”
Digit → “3”
Digit → “4”
Digit → “5”
Digit → “6”
Digit → “7”
Digit → “8”
Digit → “9”

decimal numbers

morphology

G = (N, Σ, P, S)

Num → Digit Num
Num → Digit
Digit → "0"
Digit → "1"
Digit → "2"
Digit → "3"
Digit → "4"
Digit → "5"
Digit → "6"
Digit → "7"
Digit → "8"
Digit → "9"

Σ: finite set of terminal symbols

decimal numbers

morphology

G = (N, Σ, P, S)

Num → Digit Num
Num → Digit
Digit → “0”
Digit → “1”
Digit → “2”
Digit → “3”
Digit → “4”
Digit → “5”
Digit → “6”
Digit → “7”
Digit → “8”
Digit → “9”

Σ: finite set of terminal symbols

N: finite set of non-terminal symbols

decimal numbers

morphology

G = (N, Σ, P, S)

Num → Digit Num
Num → Digit
Digit → “0”
Digit → “1”
Digit → “2”
Digit → “3”
Digit → “4”
Digit → “5”
Digit → “6”
Digit → “7”
Digit → “8”
Digit → “9”

Σ: finite set of terminal symbols

N: finite set of non-terminal symbols

S∈N: start symbol

decimal numbers

morphology

$G = (N, \Sigma, P, S)$

Num → Digit Num
Num → Digit
Digit → “0”
Digit → “1”
Digit → “2”
Digit → “3”
Digit → “4”
Digit → “5”
Digit → “6”
Digit → “7”
Digit → “8”
Digit → “9”

Σ: finite set of terminal symbols

N: finite set of non-terminal symbols

$S \in N$: start symbol

$P \subseteq N \times (N \cup \Sigma)^*$: set of production rules

decimal numbers

production

Num ⇒	Num → Digit Num
Digit Num ⇒	Digit → “4”
4 Num ⇒	Num → Digit Num
4 Digit Num ⇒	Digit → “3”
4 3 Num ⇒	Num → Digit Num
4 3 Digit Num ⇒	Digit → “0”
4 3 0 Num ⇒	Num → Digit
4 3 0 Digit ⇒	Digit → “3”
4 3 0 3	

decimal numbers

production

Num ⇒	Num → Digit Num
Digit Num ⇒	Digit → “4”
4 Num ⇒	Num → Digit Num
4 Digit Num ⇒	Digit → “3”
4 3 Num ⇒	Num → Digit Num
4 3 Digit Num ⇒	Digit → “0”
4 3 0 Num ⇒	Num → Digit
4 3 0 Digit ⇒	Digit → “3”
4 3 0 3	leftmost derivation

decimal numbers

production

Num ⇒	Num → Digit Num
Digit Num ⇒	Num → Digit Num
Digit Digit Num ⇒	Num → Digit Num
Digit Digit Digit Num ⇒	Num → Digit
Digit Digit Digit Digit ⇒	Digit → “3”
Digit Digit Digit 3 ⇒	Digit → “0”
Digit Digit 0 3 ⇒	Digit → “3”
Digit 3 0 3 ⇒	Digit → “4”
4 3 0 3	

decimal numbers

production

Num ⇒	Num → Digit Num
Digit Num ⇒	Num → Digit Num
Digit Digit Num ⇒	Num → Digit Num
Digit Digit Digit Num ⇒	Num → Digit
Digit Digit Digit Digit ⇒	Digit → “3”
Digit Digit Digit 3 ⇒	Digit → “0”
Digit Digit 0 3 ⇒	Digit → “3”
Digit 3 0 3 ⇒	Digit → “4”
4 3 0 3	rightmost derivation

binary expressions

syntax

G = (N, Σ, P, S)

Exp → Num
Exp → Exp "+" Exp
Exp → Exp "–" Exp
Exp → Exp "*" Exp
Exp → Exp "/" Exp
Exp → "(" Exp ")"

binary expressions

syntax

G = (N, Σ, P, S)

Exp → Num
Exp → Exp "+" Exp
Exp → Exp "−" Exp
Exp → Exp "*" Exp
Exp → Exp "/" Exp
Exp → "(" Exp ")"

Σ: finite set of terminal symbols

binary expressions

syntax

G = (N, Σ, P, S)

Exp → Num
Exp → Exp “+” Exp
Exp → Exp “−” Exp
Exp → Exp “*” Exp
Exp → Exp “/” Exp
Exp → “(” Exp “)”

Σ: finite set of terminal symbols

N: finite set of non-terminal symbols

binary expressions

syntax

G = (N, Σ, P, S)

Exp → Num
Exp → Exp "+" Exp
Exp → Exp "−" Exp
Exp → Exp "*" Exp
Exp → Exp "/" Exp
Exp → "(" Exp ")"

Σ: finite set of terminal symbols

N: finite set of non-terminal symbols

S∈N: start symbol

binary expressions

syntax

$G = (N, \Sigma, P, S)$

Exp → Num
Exp → Exp “+” Exp
Exp → Exp “–” Exp
Exp → Exp “*” Exp
Exp → Exp “/” Exp
Exp → “(” Exp “)”

Σ: finite set of terminal symbols

N: finite set of non-terminal symbols

$S \in N$: start symbol

$P \subseteq N \times (N \cup \Sigma)^*$: set of production rules

binary expressions

production

Exp ⇒	Exp → Exp "+" Exp
Exp + Exp ⇒	Exp → Exp "*" Exp
Exp + Exp * Exp ⇒	Exp → Num
3 + Exp * Exp ⇒	Exp → Num
3 + 4 * Exp ⇒	Exp → Num
3 + 4 * 5	

formal grammar $G = (N, \Sigma, P, S)$

nonterminal symbols N

terminal symbols Σ

production rules $P \subseteq (N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$

start symbol $S \in N$

formal grammar $G = (N, \Sigma, P, S)$

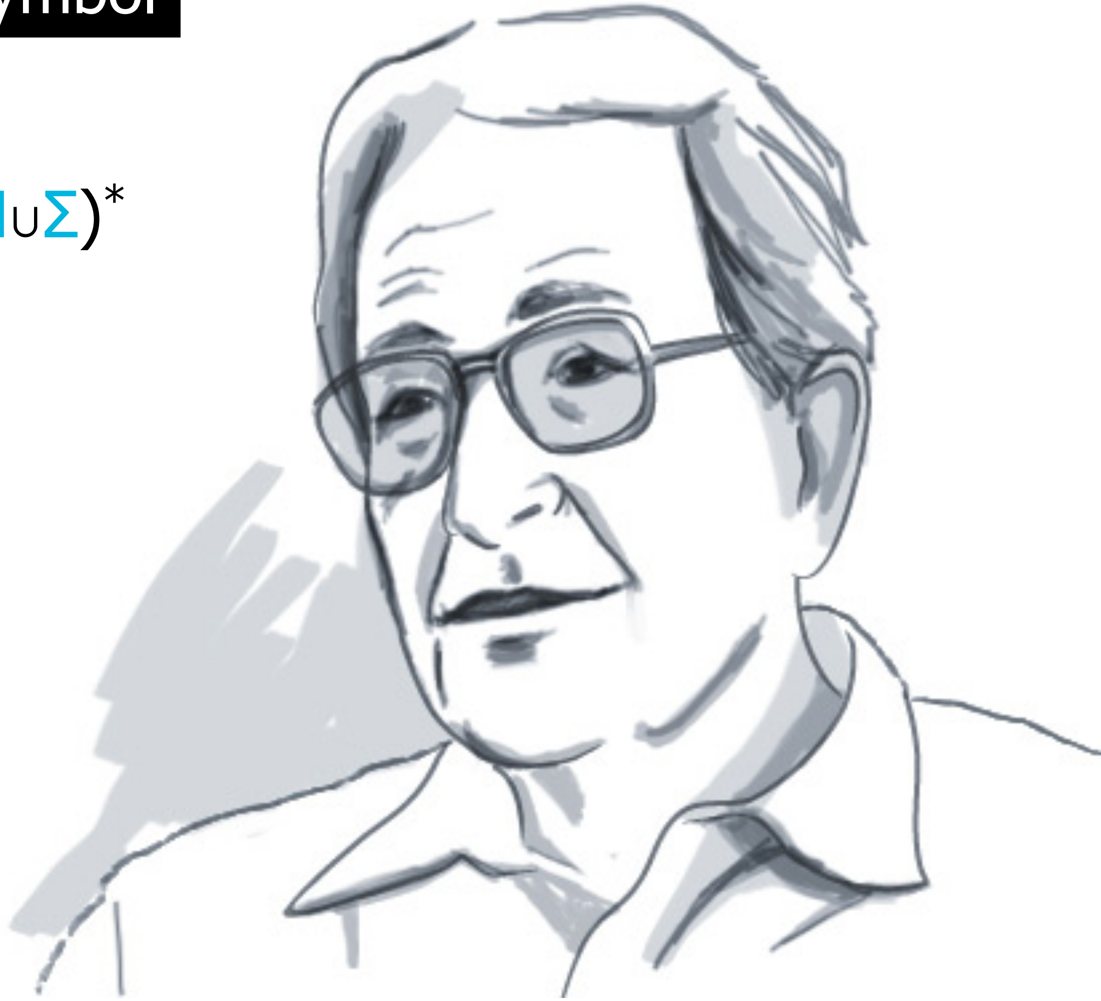
nonterminal symbols N

nonterminal symbol

terminal symbols Σ

production rules $P \subseteq (N \cup \Sigma)^* \boxed{N} (N \cup \Sigma)^* \times (N \cup \Sigma)^*$

start symbol $S \in N$



formal grammar $G = (N, \Sigma, P, S)$

nonterminal symbols N

terminal symbols Σ

production rules $P \subseteq \boxed{(N\cup\Sigma)^*}\, N\, \boxed{(N\cup\Sigma)^*} \times (N\cup\Sigma)^*$

start symbol $S\in N$

context

formal grammar $G = (N, \Sigma, P, S)$

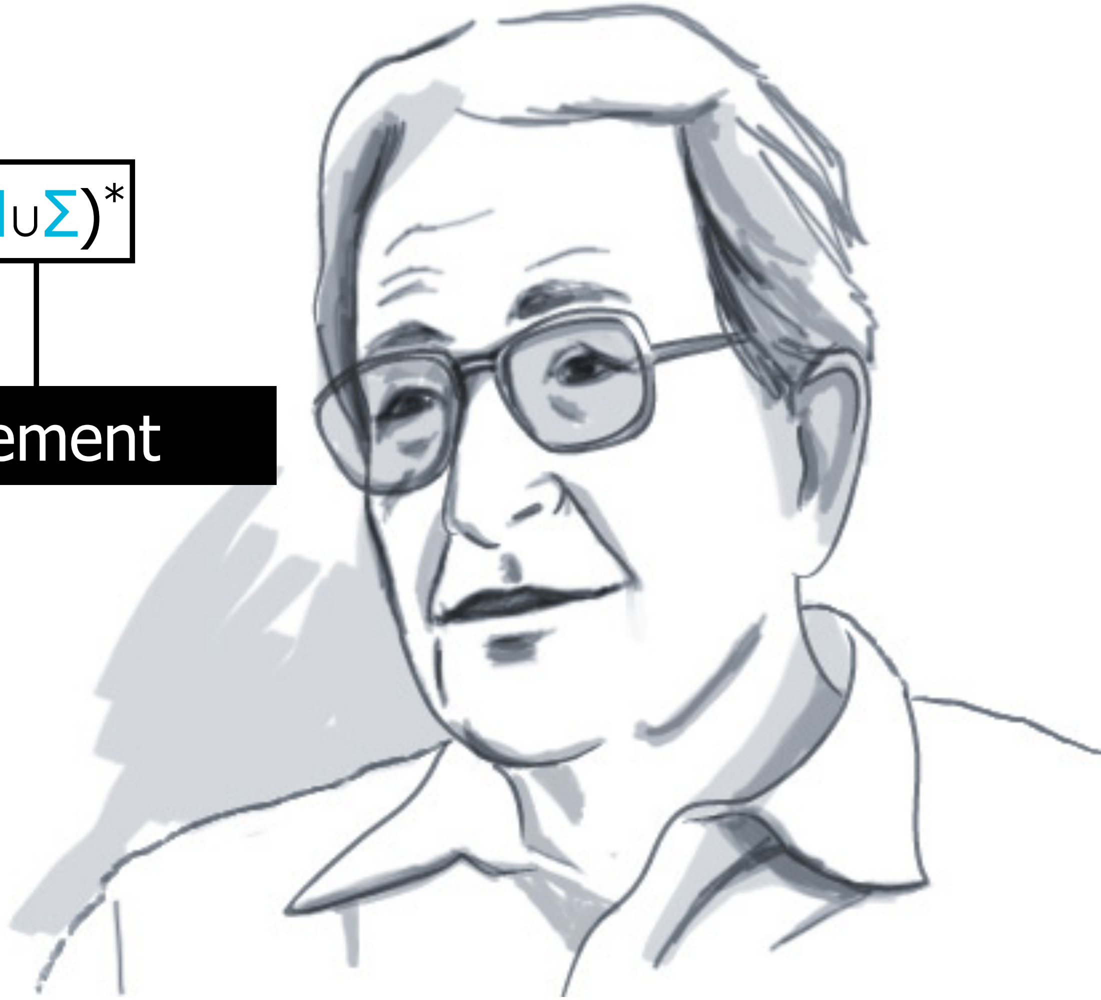
nonterminal symbols N

terminal symbols Σ

production rules $P \subseteq (N \cup \Sigma)^* \, N \, (N \cup \Sigma)^* \times \boxed{(N \cup \Sigma)^*}$

start symbol $S \in N$

replacement



formal grammar $G = (N, \Sigma, P, S)$

nonterminal symbols N

terminal symbols Σ

production rules $P \subseteq (N \cup \Sigma)^* \, N \, (N \cup \Sigma)^* \times (N \cup \Sigma)^*$

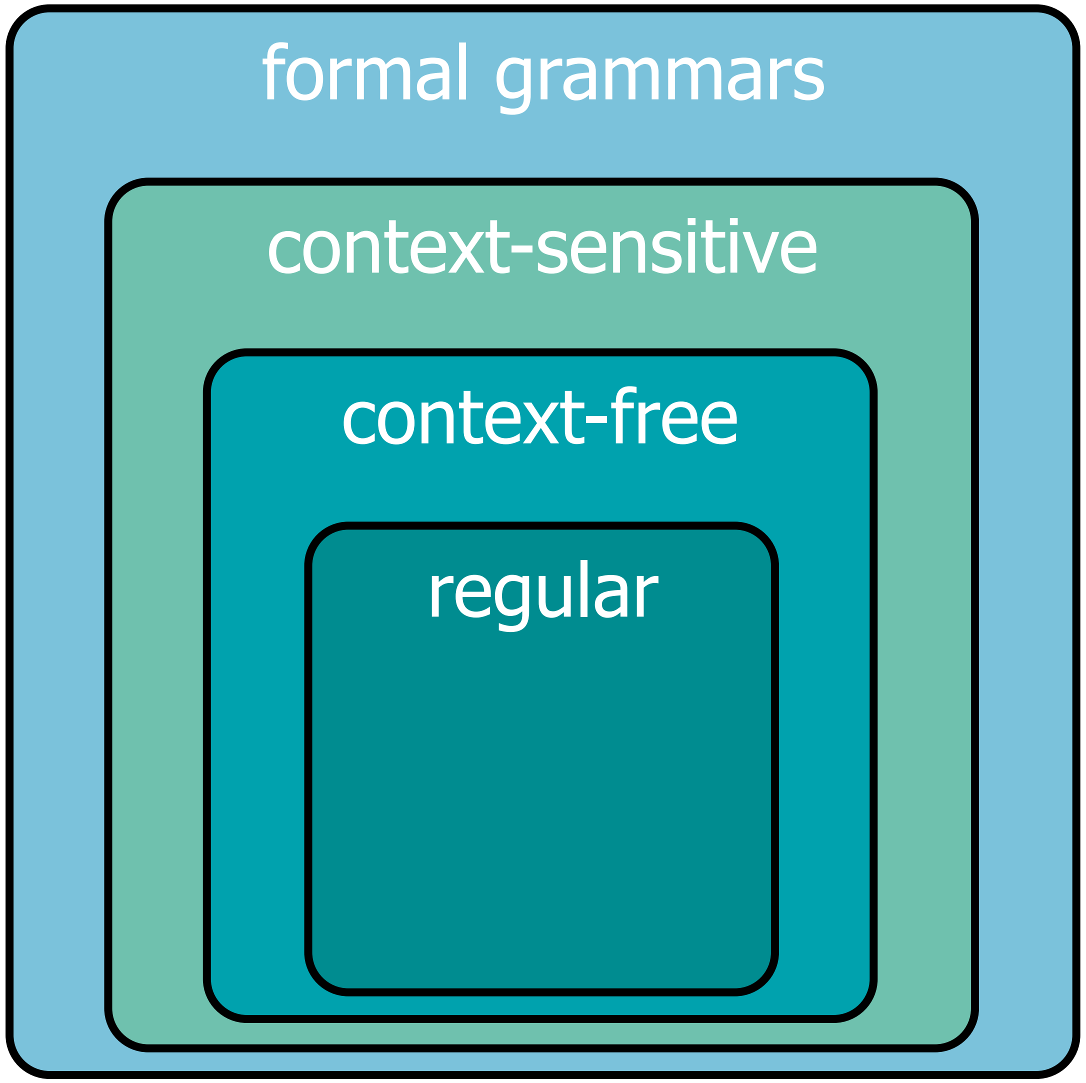
start symbol $S \in N$

type-0, unrestricted: $P \subseteq (N \cup \Sigma)^* \, N \, (N \cup \Sigma)^* \times (N \cup \Sigma)^*$

type-1, context-sensitive: (a A c, a b c)

type-2, context-free: $P \subseteq N \times (N \cup \Sigma)^*$

type-3, regular: (A, x) or (A, xB)



formal grammars

context-sensitive

context-free

regular

formal grammar G

derivation relation $\Rightarrow_G$

formal language $L(G) \subseteq \Sigma^*$

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$$

formal grammar $G = (N, \Sigma, P, S)$

derivation relation $\Rightarrow_G \subseteq (N \cup \Sigma)^* \times (N \cup \Sigma)^*$

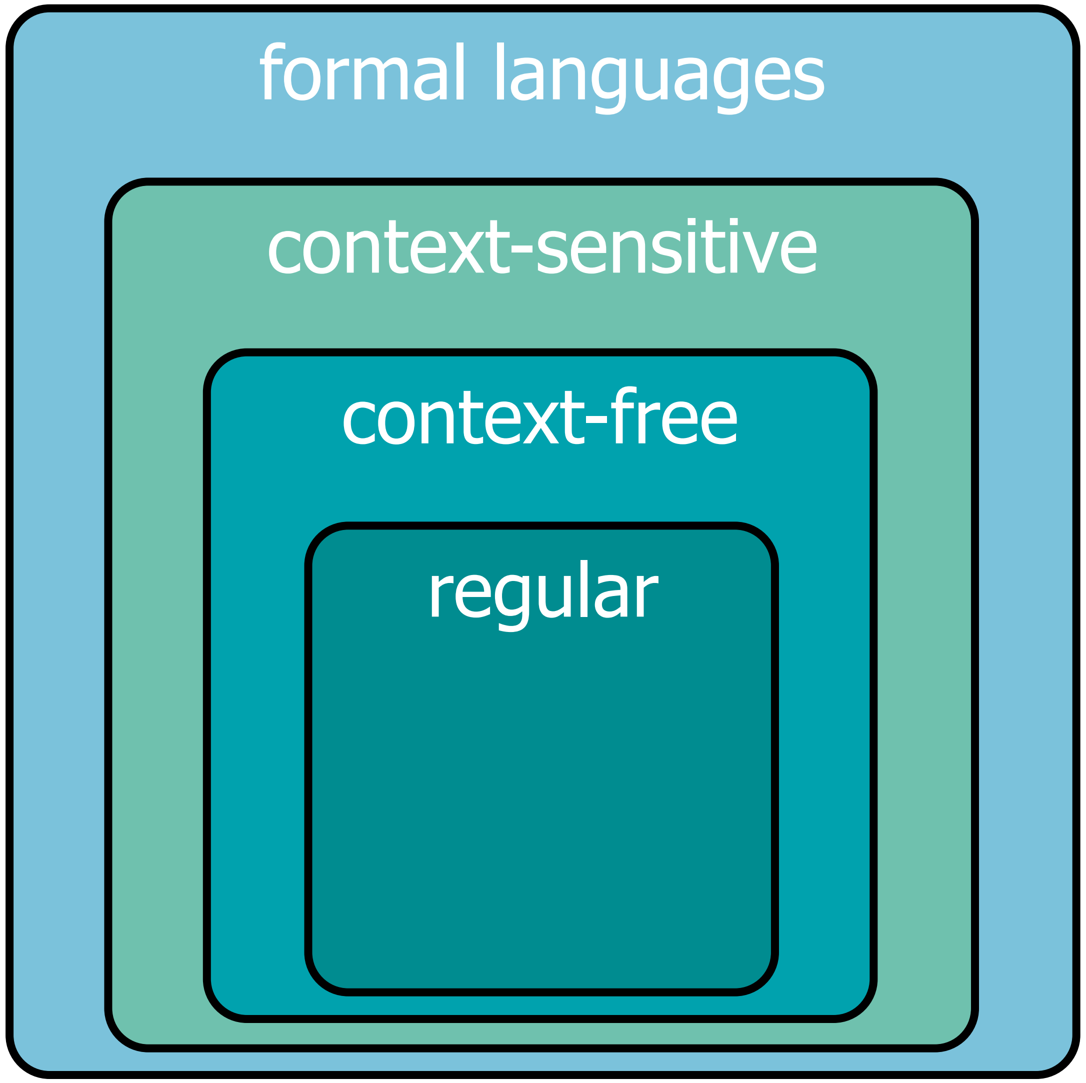
$w \Rightarrow_G w' \Leftrightarrow$

$\exists (p, q) \in P: \exists u,v \in (N \cup \Sigma)^*:$

$w = u\ p\ v \wedge w' = u\ q\ v$

formal language $L(G) \subseteq \Sigma^*$

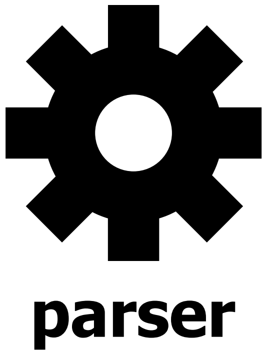
$L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$

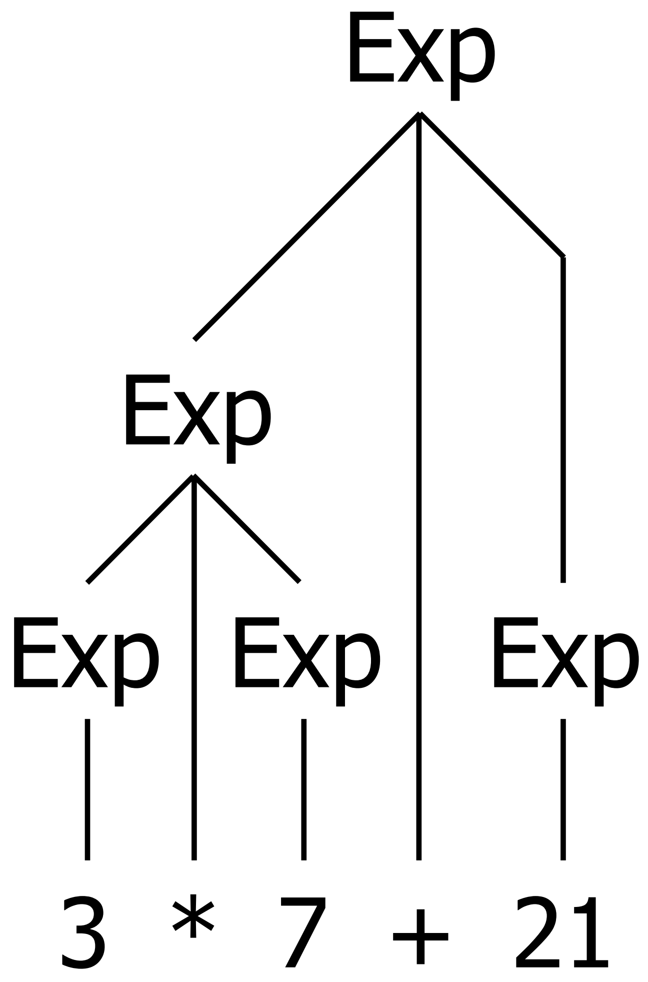

formal languages
context-sensitive
context-free
regular

Derivation is about productivity.

But what about parsing?

3 * 7 + 21





word problem

theoretical computer science

decidability & complexity

word problem $\chi_L: \Sigma^* \rightarrow \{0,1\}$

$w \rightarrow 1$, if $w \in L$

$w \rightarrow 0$, else

theoretical computer science

decidability & complexity

word problem χ_L: $\Sigma^* \rightarrow \{0,1\}$

$w \rightarrow 1$, if $w \in L$

$w \rightarrow 0$, else

decidability

type-0: semi-decidable

type-1, type-2, type-3: decidable

theoretical computer science

decidability & complexity

word problem $\chi_L: \Sigma^* \rightarrow \{0,1\}$

$w \rightarrow 1$, if $w \in L$

$w \rightarrow 0$, else

decidability

type-0: semi-decidable

type-1, type-2, type-3: decidable

complexity

type-1: PSPACE-complete

type-2, type-3: P

theoretical computer science

decidability & complexity

word problem $\chi_L: \Sigma^* \rightarrow \{0,1\}$

- $w \rightarrow 1$, if $w \in L$
- $w \rightarrow 0$, else

decidability

- type-0: semi-decidable
- type-1, type-2, type-3: decidable

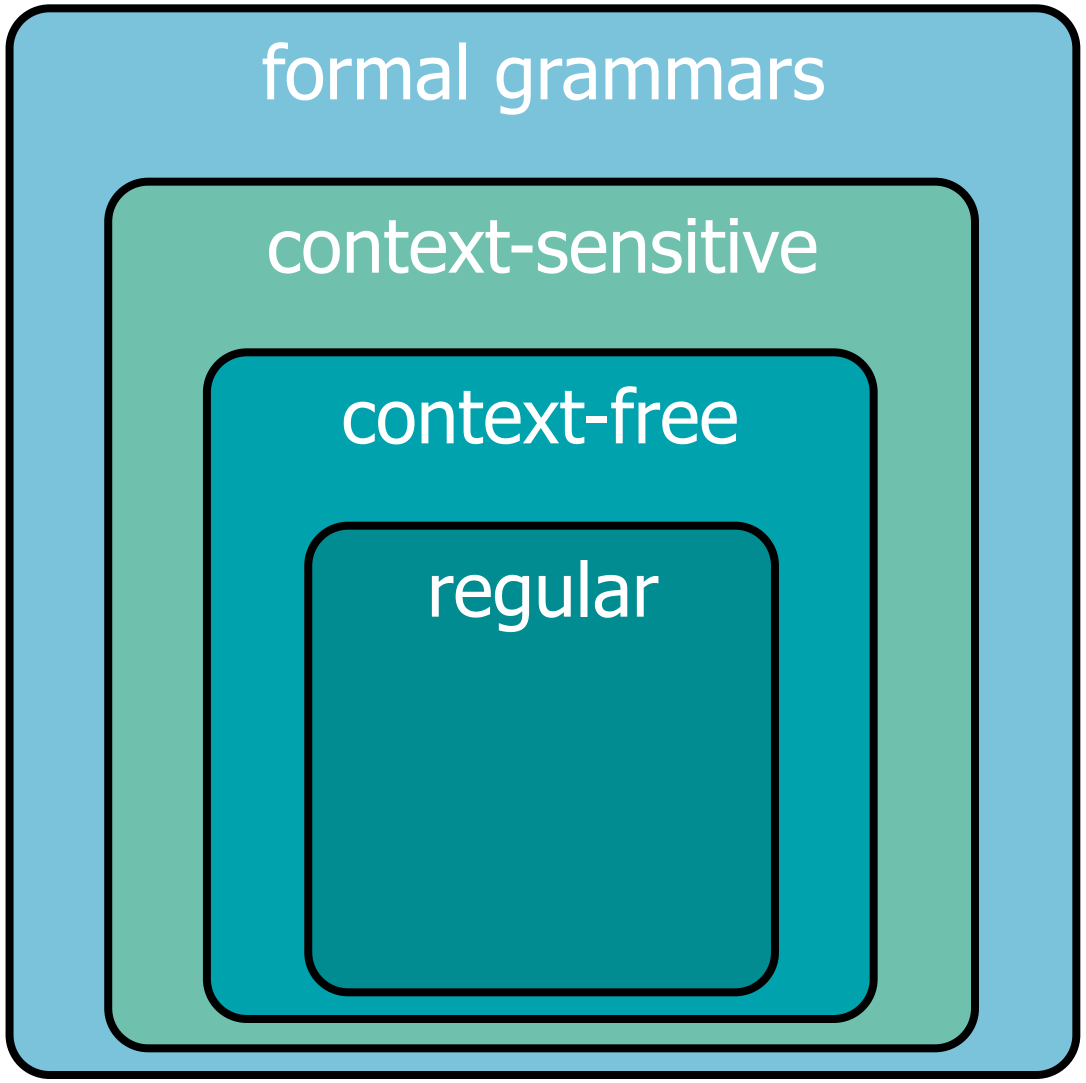
complexity

- type-1: PSPACE-complete
- type-2, type-3: P

$\boxed{\text{PSPACE} \supseteq \text{NP} \supseteq \text{P}}$

theoretical computer science

decidability & complexity



formal grammars

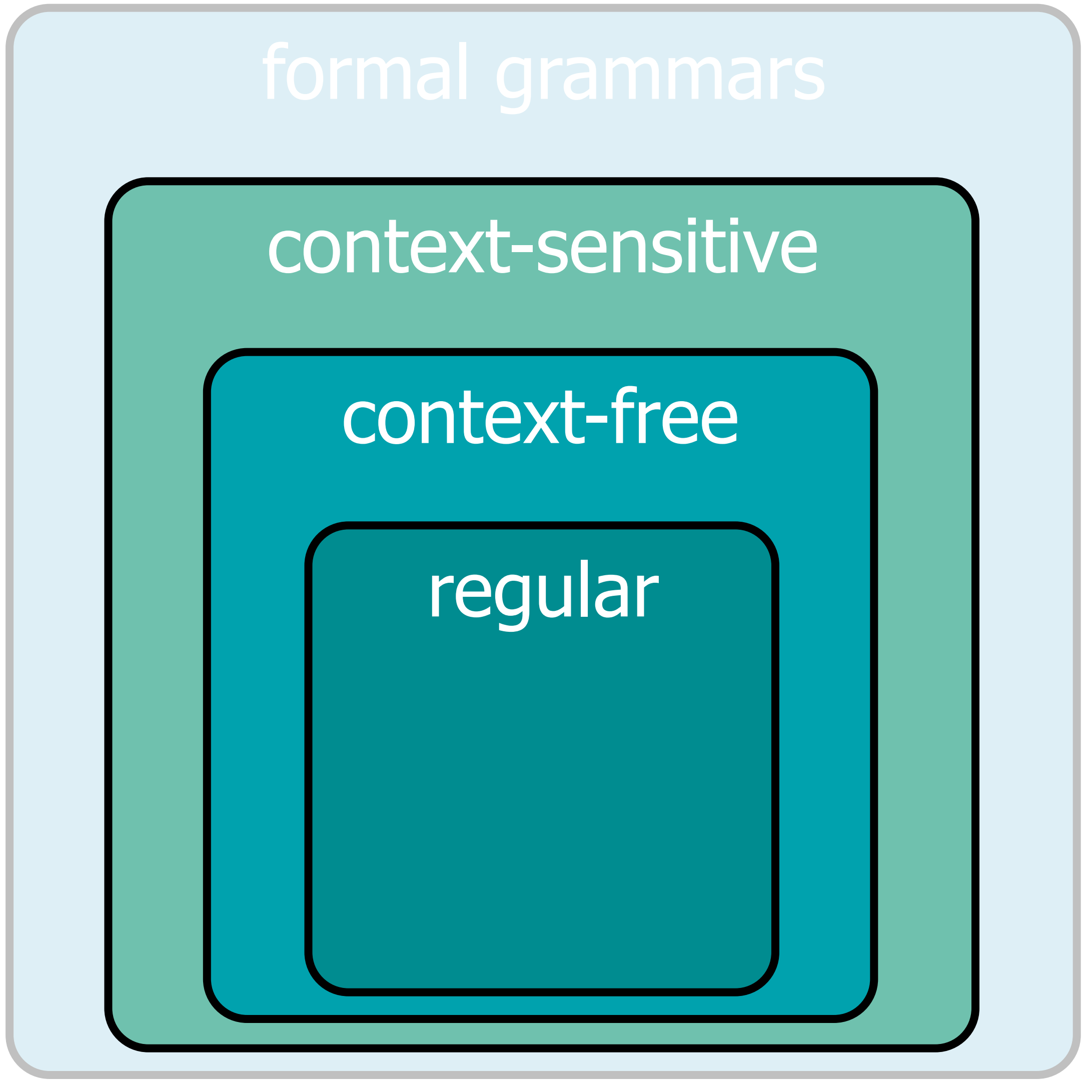
context-sensitive

context-free

regular

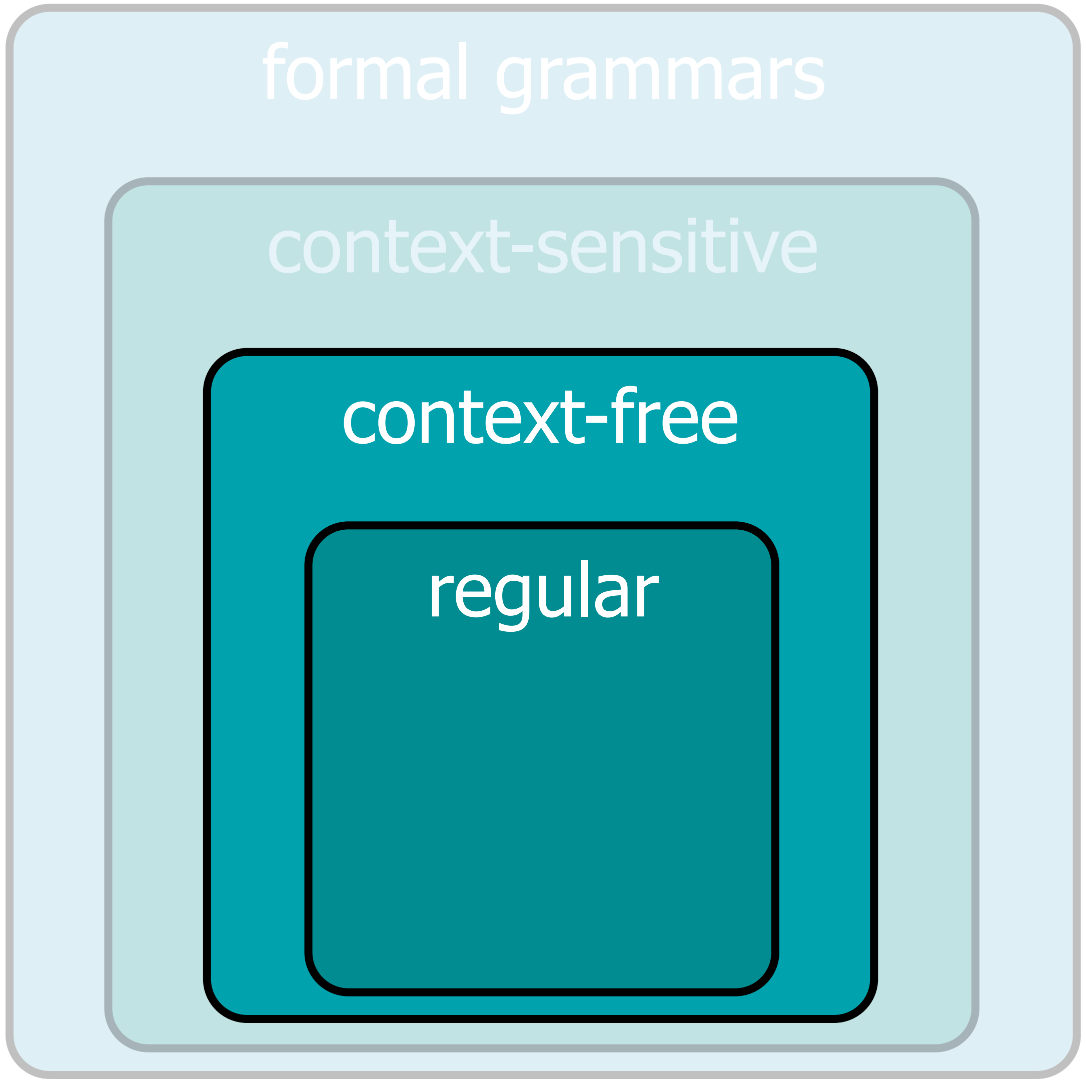
theoretical computer science

decidability & complexity



theoretical computer science

decidability & complexity



context-free grammars

production vs. reduction rules

Exp → Num
Exp → Exp “+” Exp
Exp → Exp “−” Exp
Exp → Exp “*” Exp
Exp → Exp “/” Exp
Exp → “(” Exp “)”

context-free grammars

production vs. reduction rules

Exp → Num
Exp → Exp “+” Exp
Exp → Exp “–” Exp
Exp → Exp “*” Exp
Exp → Exp “/” Exp
Exp → “(” Exp “)”

productive form

context-free grammars

production vs. reduction rules

Exp → Num
Exp → Exp “+” Exp
Exp → Exp “–” Exp
Exp → Exp “*” Exp
Exp → Exp “/” Exp
Exp → “(” Exp “)”

Num → Exp
Exp “+” Exp → Exp
Exp “–” Exp → Exp
Exp “*” Exp → Exp
Exp “/” Exp → Exp
“(” Exp “)” → Exp

productive form

context-free grammars

production vs. reduction rules

Exp → Num
Exp → Exp “+” Exp
Exp → Exp “−” Exp
Exp → Exp “*” Exp
Exp → Exp “/” Exp
Exp → “(” Exp “)”

productive form

Num → Exp
Exp “+” Exp → Exp
Exp “−” Exp → Exp
Exp “*” Exp → Exp
Exp “/” Exp → Exp
“(” Exp “)” → Exp

reductive form

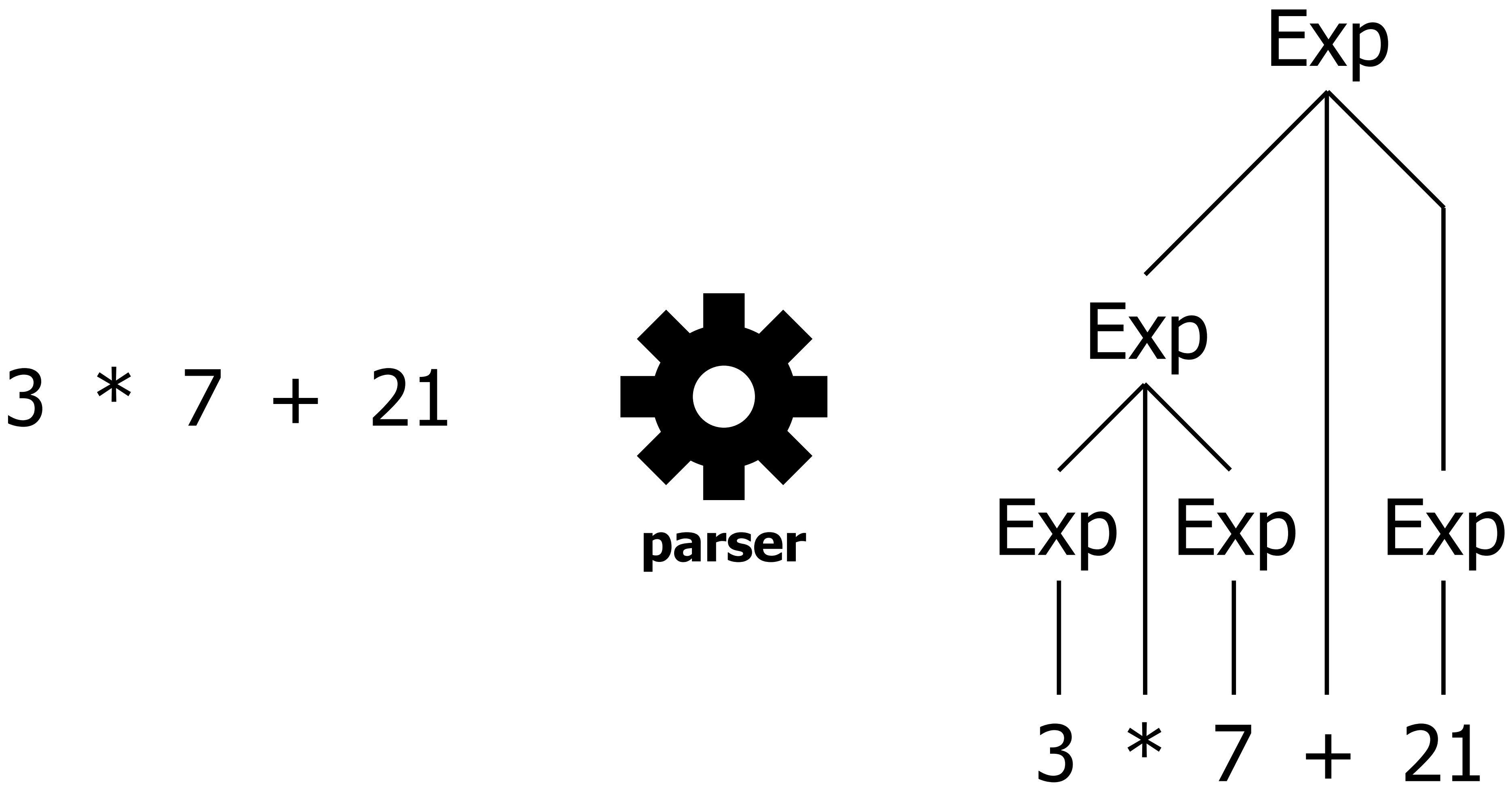
binary expressions

reduction

3 + 4 * 5 ⇒	Num	→ Exp
Exp + 4 * 5 ⇒	Num	→ Exp
Exp + Exp * 5 ⇒	Num	→ Exp
Exp + Exp * Exp ⇒	Exp “*” Exp	→ Exp
Exp + Exp ⇒	Exp “+” Exp	→ Exp
Exp		

The word problem is about membership.

But what about structure?

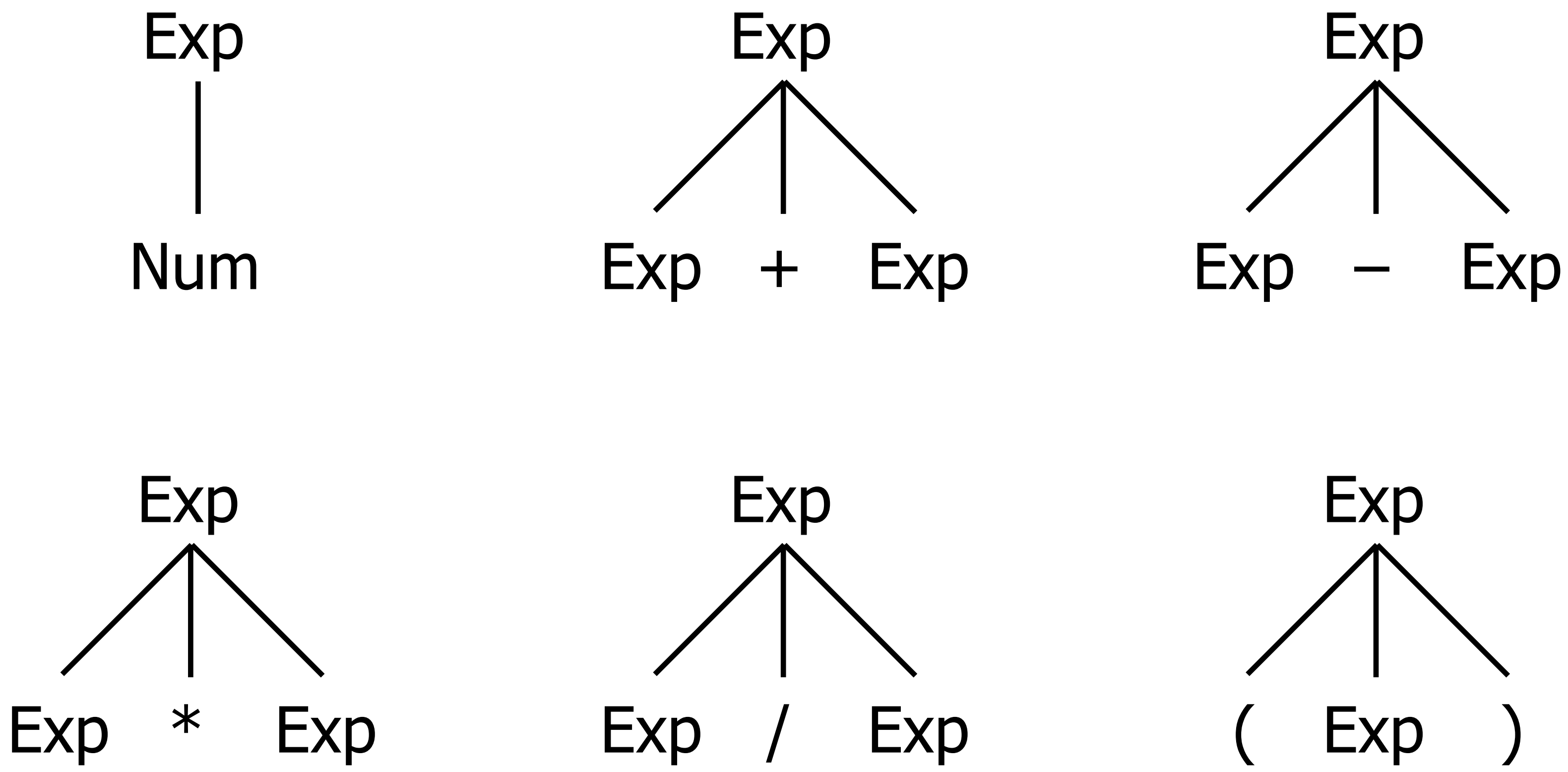

3 * 7 + 21

parser

syntax trees

context-free grammars

tree construction rules



Exp → Num

Exp → Exp + Exp

Exp → Exp – Exp

Exp → Exp * Exp

Exp → Exp / Exp

Exp → (Exp)

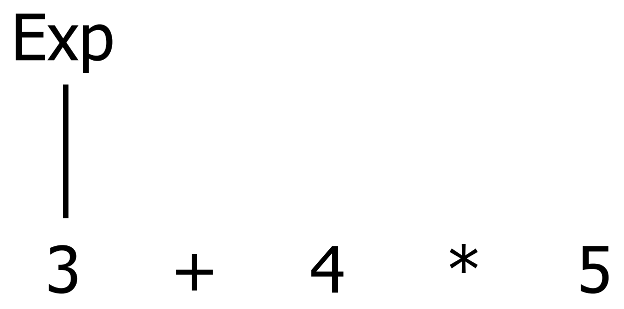
binary expressions

tree construction

3 + 4 * 5

binary expressions

tree construction

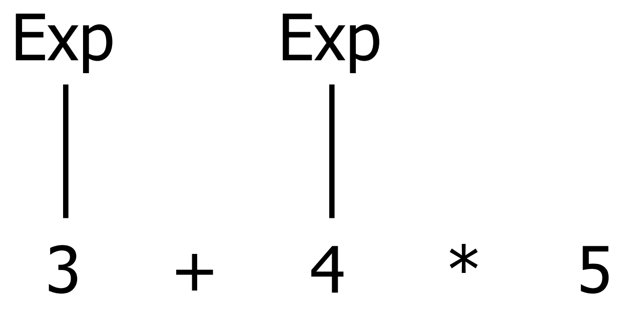


Exp

3 + 4 * 5

binary expressions

tree construction

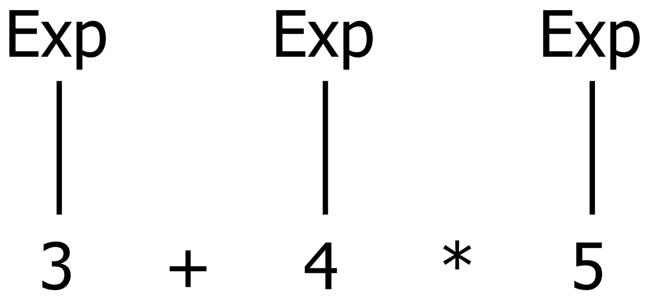


Exp Exp

3 + 4 * 5

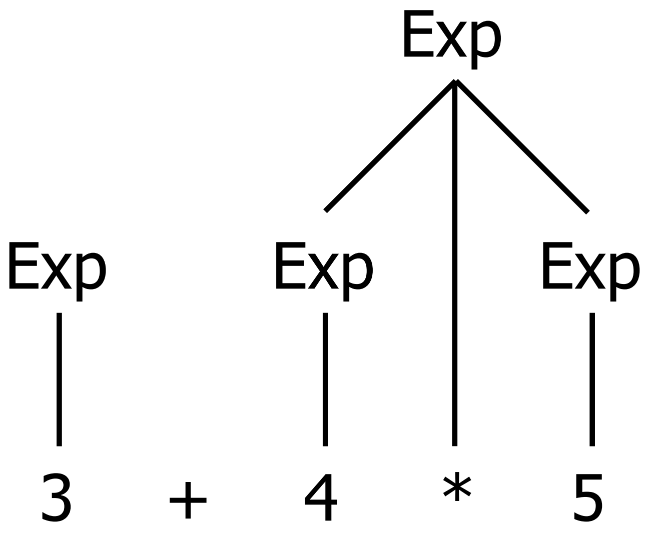
binary expressions

tree construction



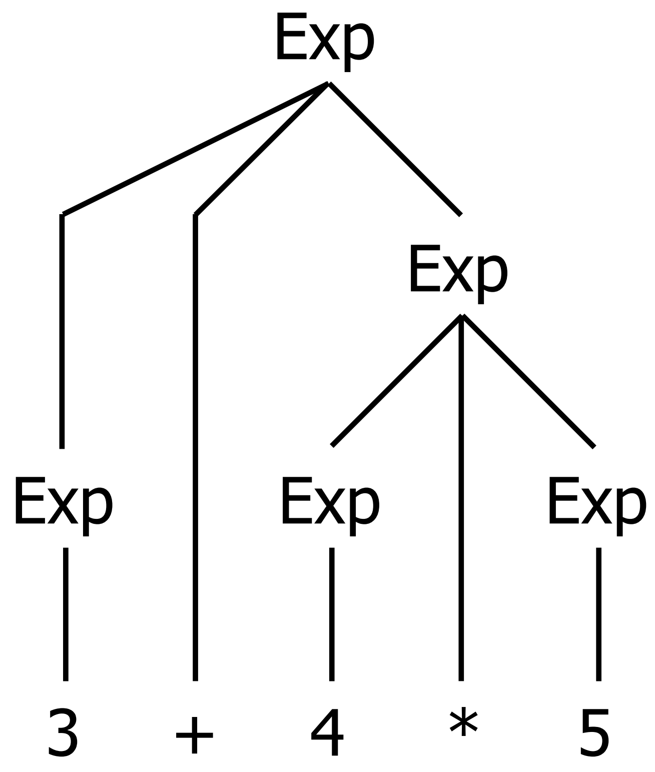
binary expressions

tree construction



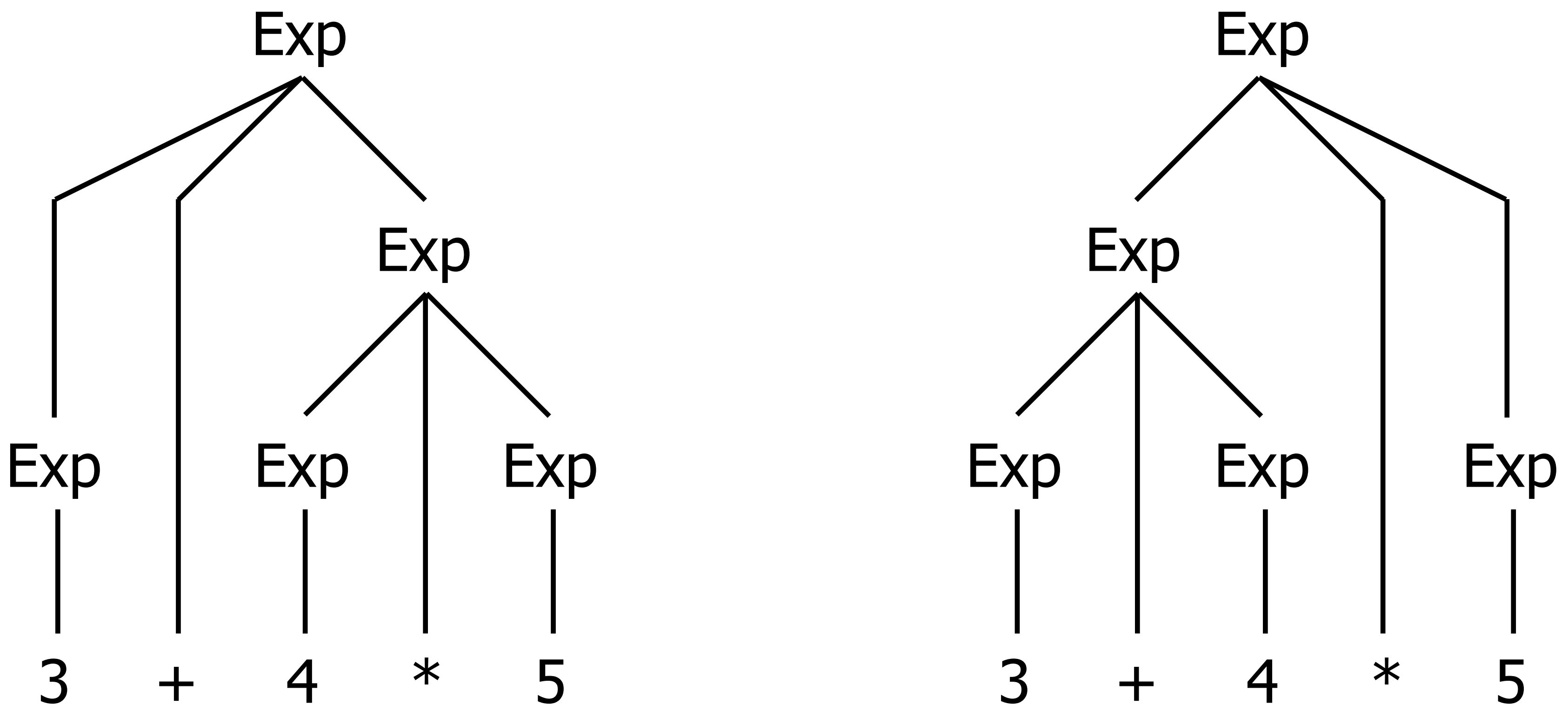
binary expressions

tree construction



binary expressions

ambiguity



context-free grammars

ambiguity

syntax trees

- **different trees** for **same sentence**

derivations

- **different leftmost derivations** for **same sentence**
- **different rightmost derivations** for **same sentence**
- **NOT** just different derivations for same sentence

syntax trees

parse trees & abstract syntax trees

parse trees

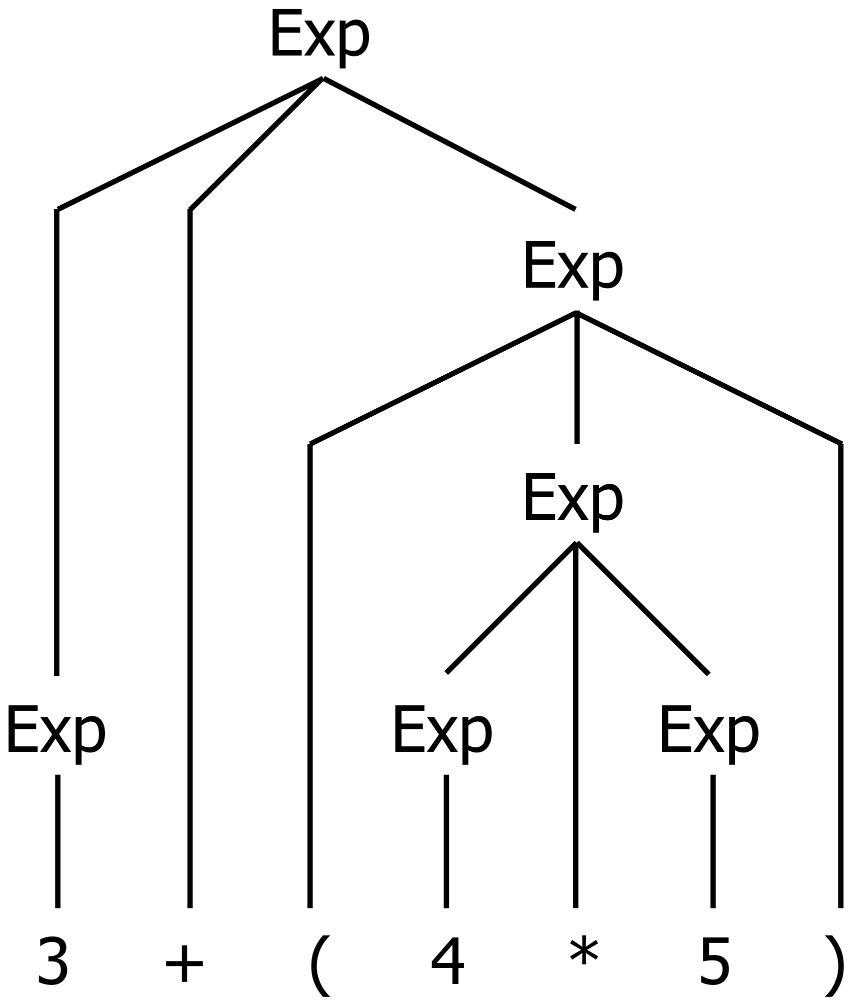
- parent node: nonterminal symbol
- child nodes: terminal symbols

abstract syntax trees (ASTs)

- abstract over terminal symbols
- convey information at parent nodes
- abstract over injective production rules

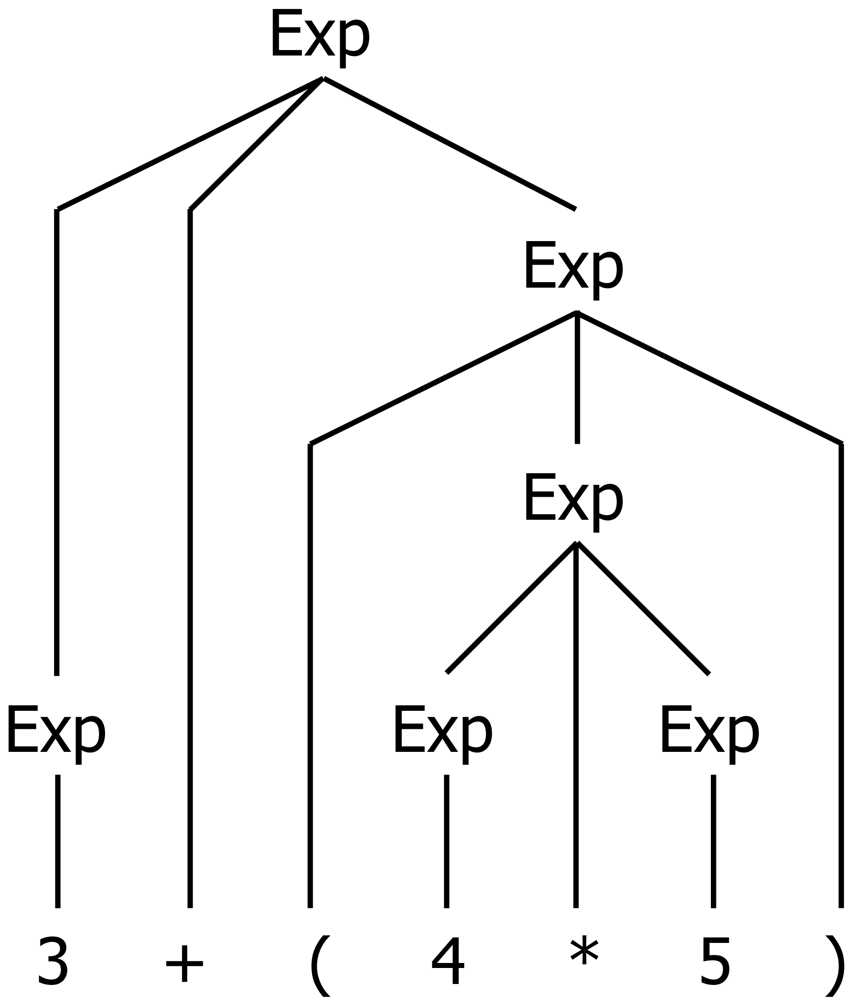
binary expressions

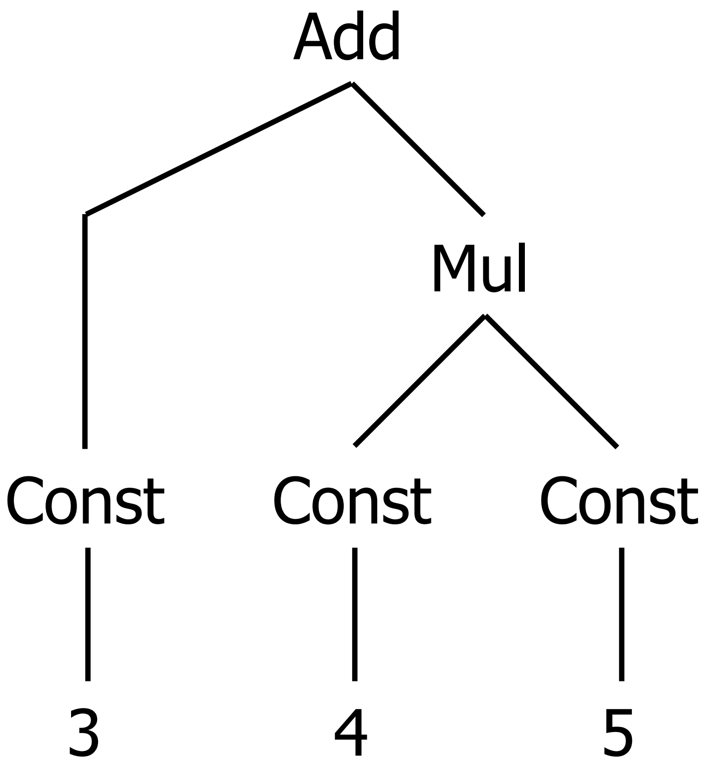
parse tree & abstract syntax tree



binary expressions

parse tree & abstract syntax tree





attribution

slide	title	author	license
1	The Pine, Saint Tropez	Paul Signac	public domain
2, 3, 35, 41	PICOL icons	Melih Bilgil	CC BY 3.0
9	Writing	Caitlin Regan	CC BY 2.0
10	Latin Grammar	Anthony Nelzin	
13, 15, 29-34, 38	Noam Chomsky	Fellowsisters	CC BY-NC-SA 2.0