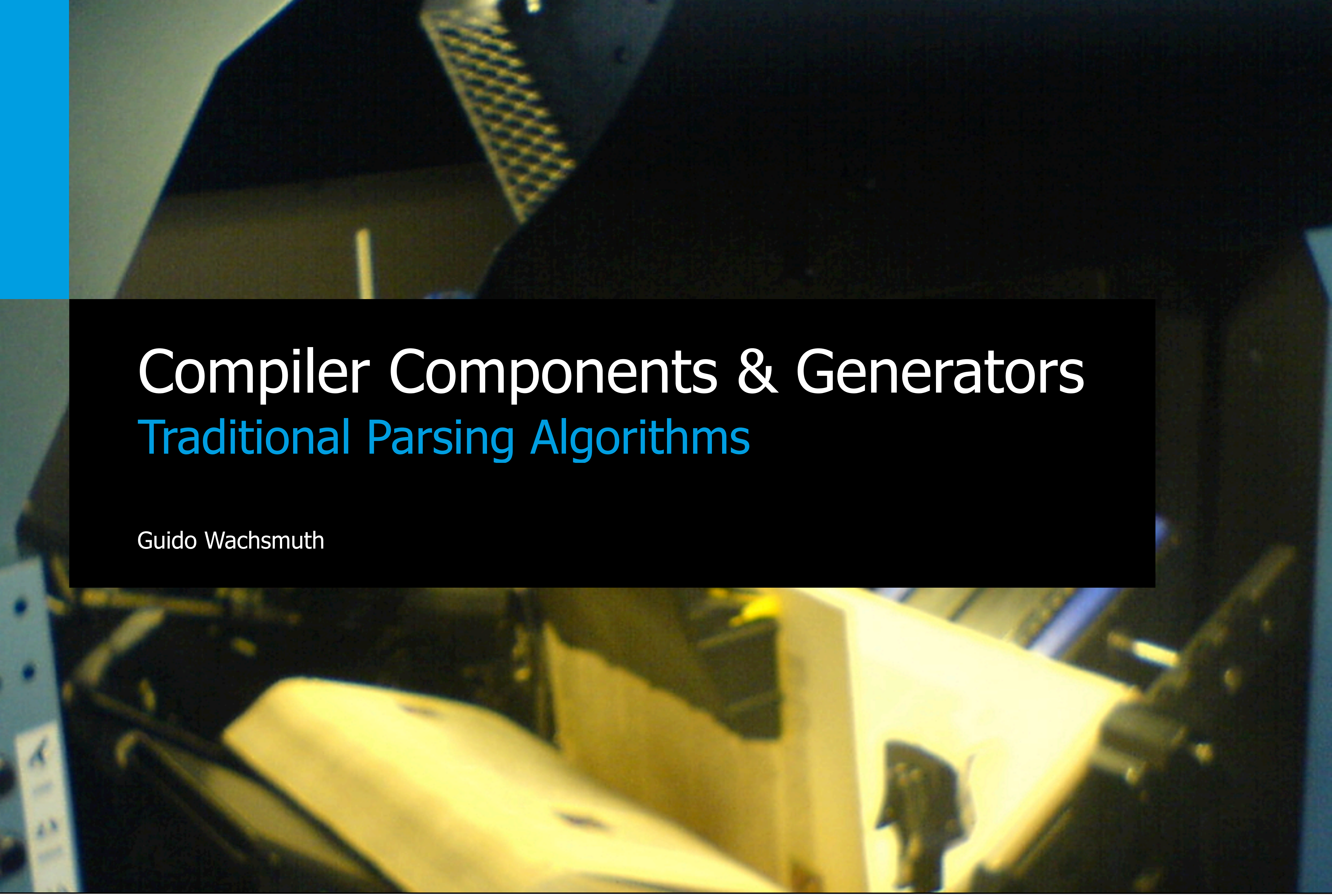


Compiler Components & Generators

Traditional Parsing Algorithms

Guido Wachsmuth

Recap: Lexical Analysis

lessons learned

What are the formalisms to describe regular languages?

- regular grammars
- regular expressions
- finite state automata

Why are these formalisms equivalent?

- constructive proofs

How can we generate compiler tools from that?

- implement DFAs
- generate transition tables

Overview

today's lecture

Overview
today's lecture

efficient parsing algorithms

- predictive parsing
- LR parsing

Overview

today's lecture

efficient parsing algorithms

- predictive parsing
- LR parsing

grammar classes

- LL(k) grammars
- LR(k) grammars

I

predictive parsing

Recap: A Theory of Language

formal languages

Recap: A Theory of Language

formal languages

vocabulary Σ

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

Recap: A Theory of Language

formal languages

vocabulary Σ

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

string over Σ

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

Recap: A Theory of Language

formal languages

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
- $\lambda \subseteq \Sigma^*$

Recap: A Theory of Language

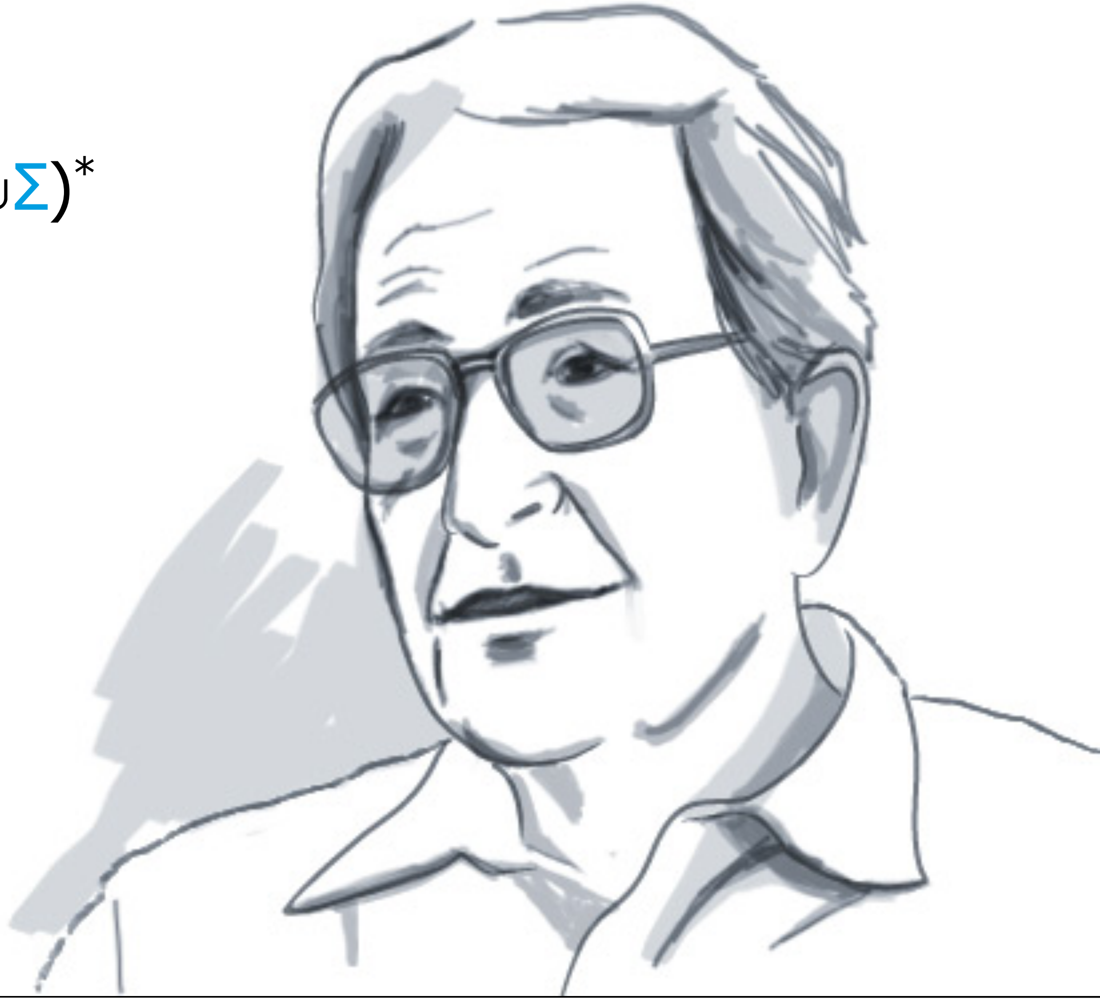
formal grammars

Recap: A Theory of Language

formal grammars

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 $S \in N$



Recap: A Theory of Language

formal grammars

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 $S \in N$

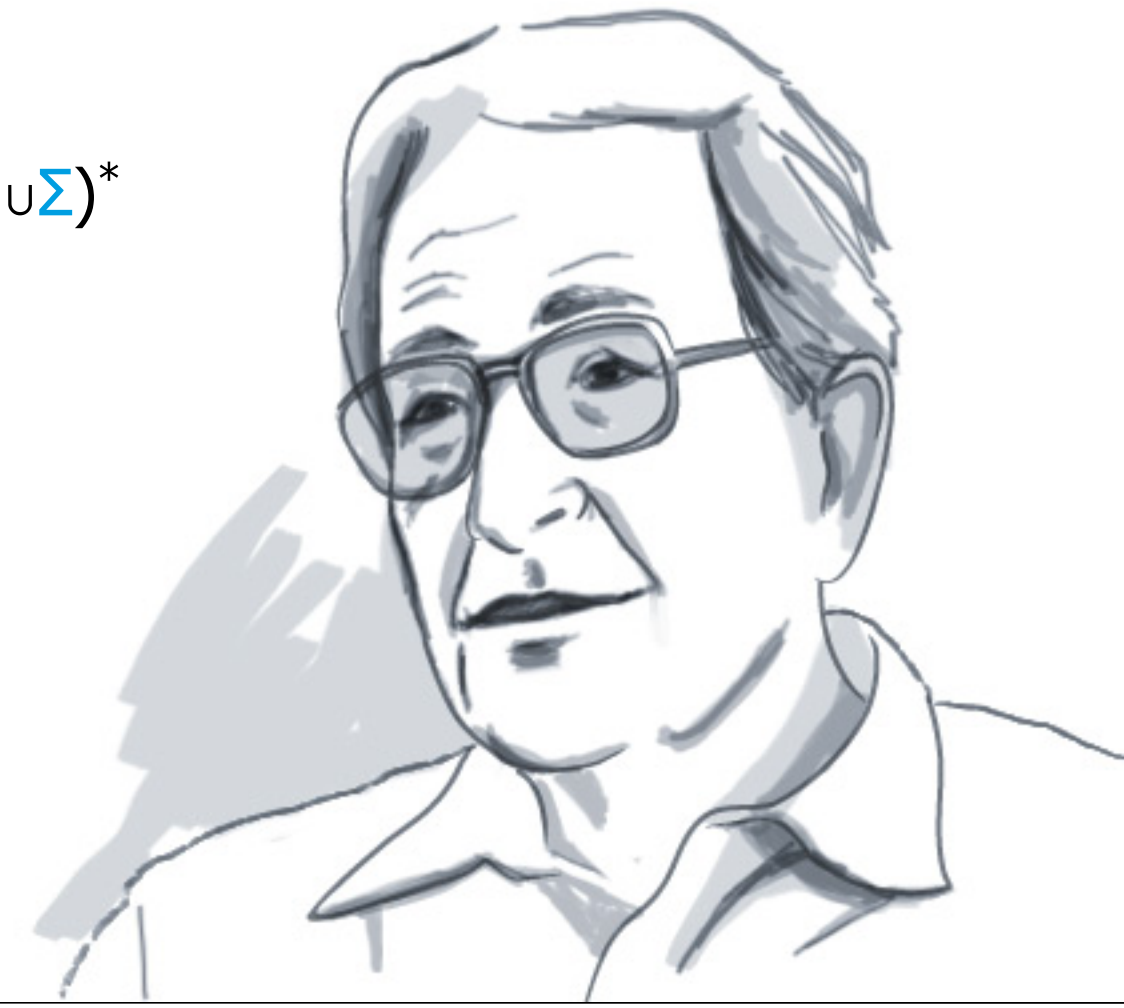
Recap: A Theory of Language

formal grammars

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



Recap: A Theory of Language

formal grammars

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

Recap: A Theory of Language

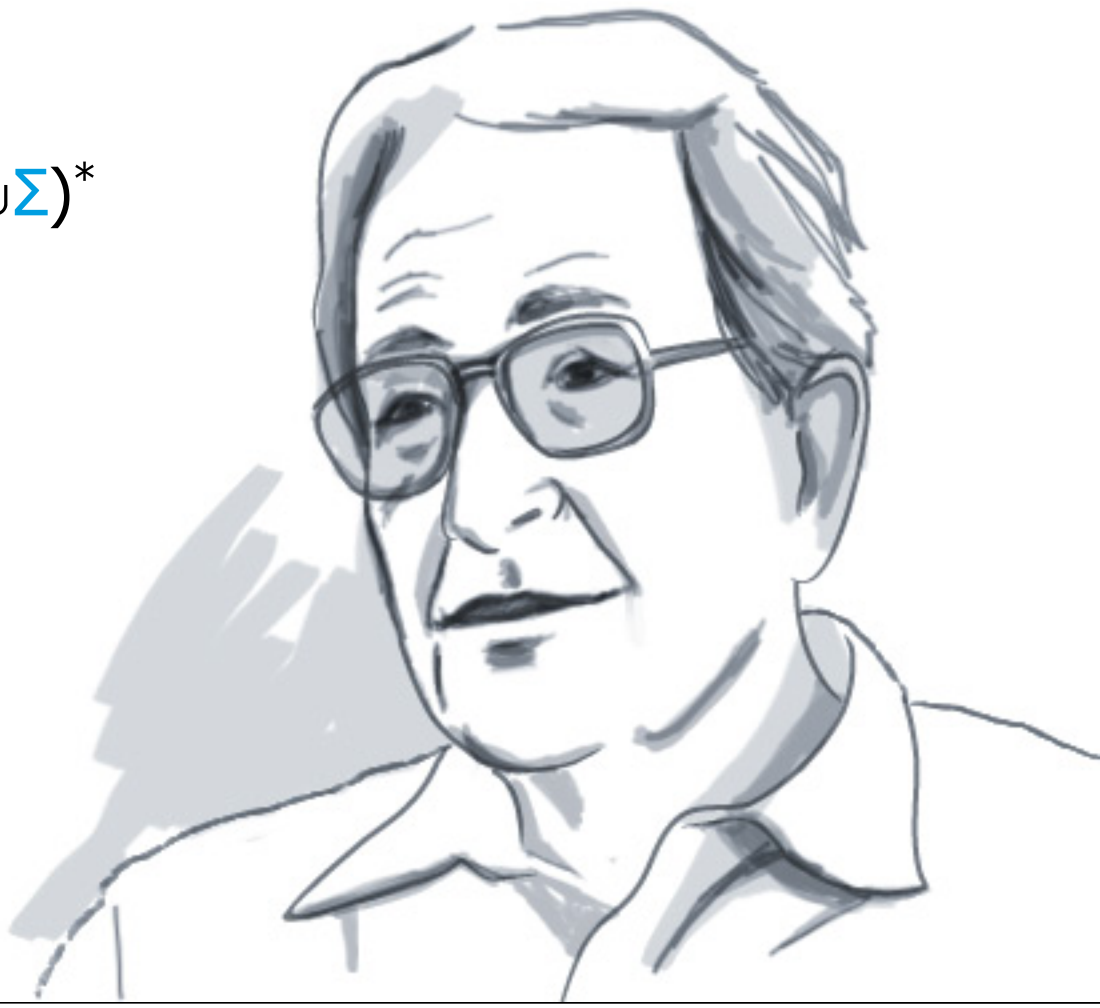
formal grammars

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 $S \in N$

grammar classes

- type-0, unrestricted
- 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)



Recap: A Theory of Language

formal languages

Recap: A Theory of Language

formal languages

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

Recap: A Theory of Language

formal languages

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

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

$w \Rightarrow_G w' \Leftrightarrow$

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

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

Recap: A Theory of Language

formal languages

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

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

Recap: A Theory of Language

formal languages

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

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

classes of formal languages

Predictive parsing

recursive descent

Exp → "while" Exp "do" Exp

```
public void parseExp() {
   consume(WHILE);
   parseExp();
   consume(DO);
   parseExp();
}
```

Predictive parsing

look ahead

Exp → "while" Exp "do" Exp

Exp → "if" Exp "then" Exp "else" Exp

```
public void parseExp() {

   switch current() {
      case WHILE: consume(WHILE); parseExp(); ...; break;
      case IF   : consume(IF); parseExp(); ...; break;
      default   : error();
   }
}
```

Predictive parsing

parse table

rows

- nonterminal symbols N
- symbol to parse

columns

- terminal symbols Σ^k
- look ahead k

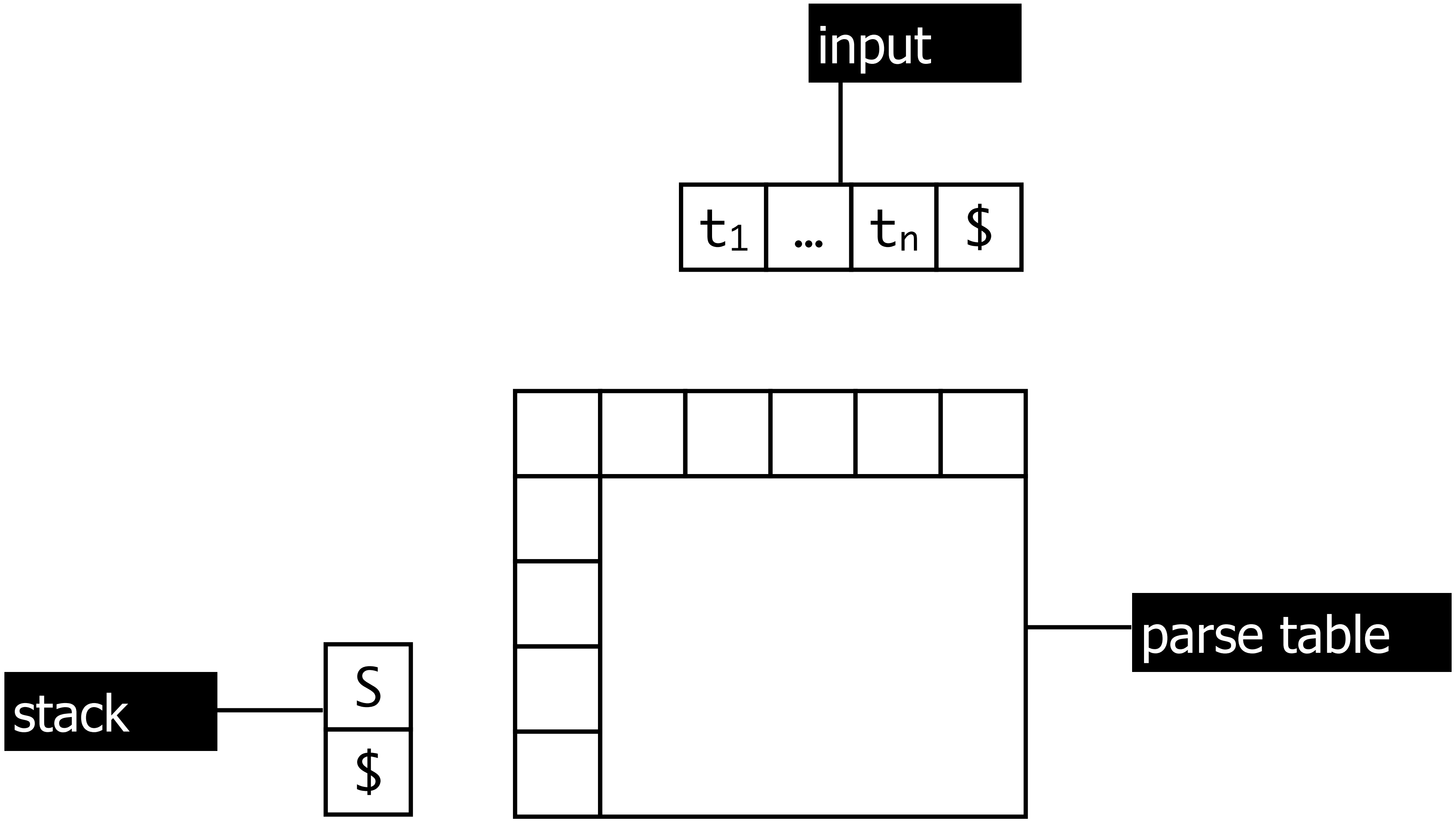
entries

- production rules P
- possible conflicts

	T_1	T_2	T_3	...
N_1	$N_1 \rightarrow$...		$N_1 \rightarrow$...	
N_2		$N_2 \rightarrow$...		
N_3		$N_3 \rightarrow$...	$N_3 \rightarrow$...	
N_4	$N_4 \rightarrow$...			
N_5		$N_5 \rightarrow$...		
N_6	$N_6 \rightarrow$...	$N_6 \rightarrow$...		
N_7			$N_7 \rightarrow$...	
N_8	$N_8 \rightarrow$...	$N_8 \rightarrow$...	$N_8 \rightarrow$...	
...				

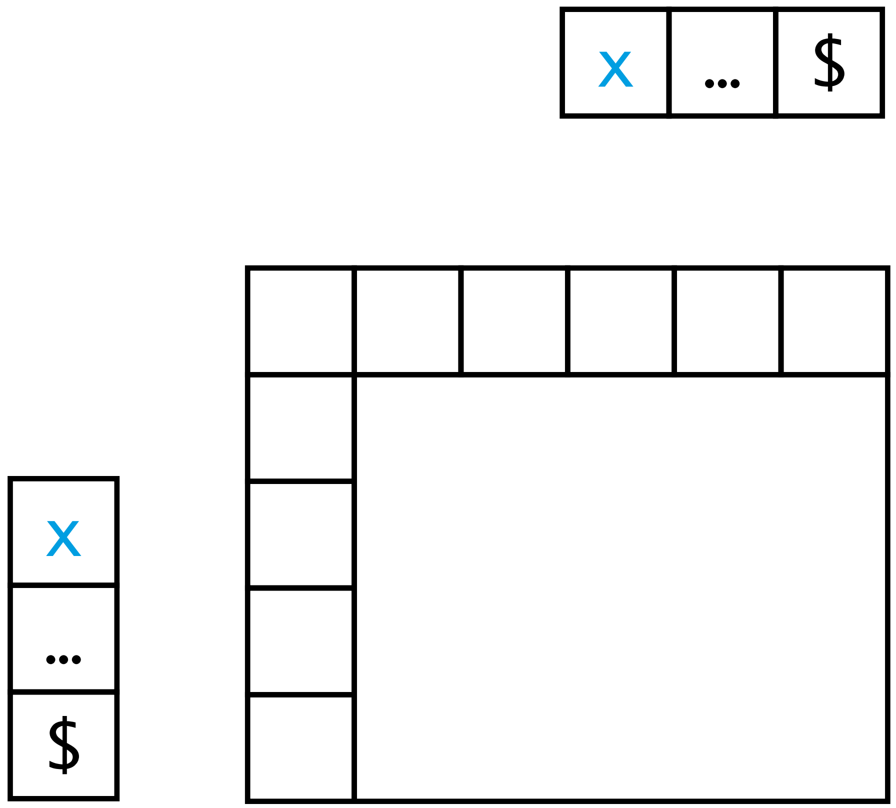
Predictive parsing

automaton

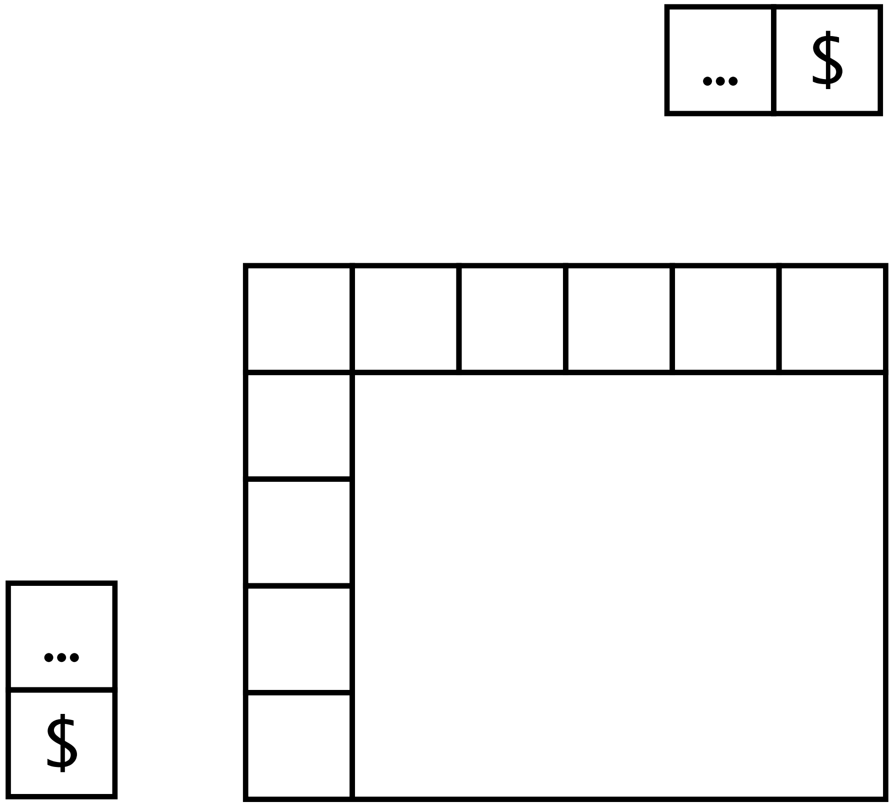


Predictive parsing

automaton

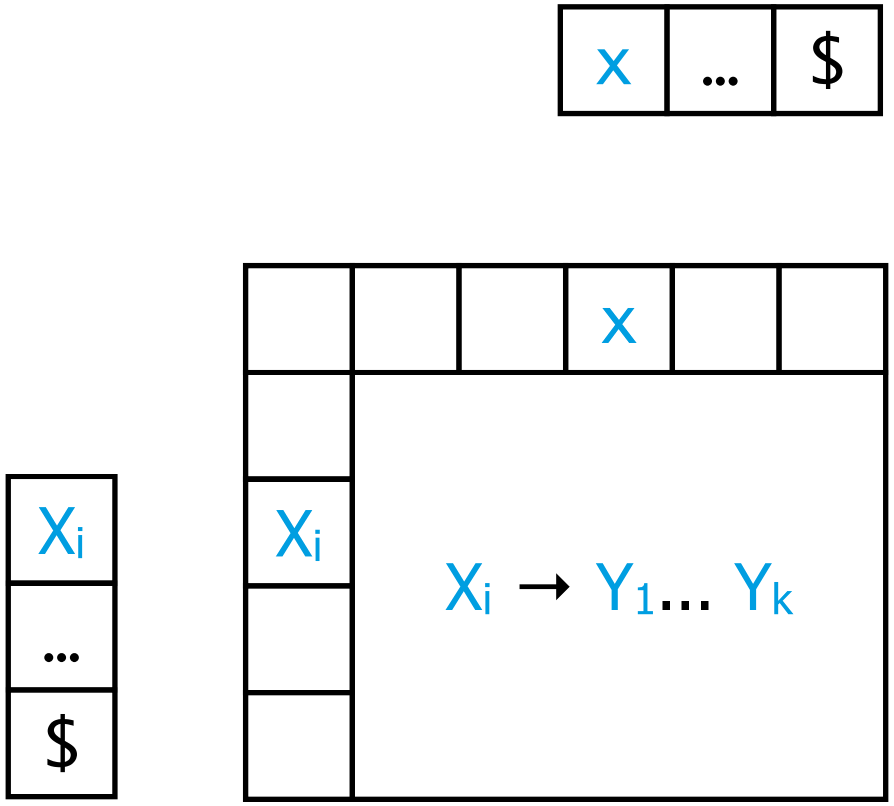


Predictive parsing
automaton



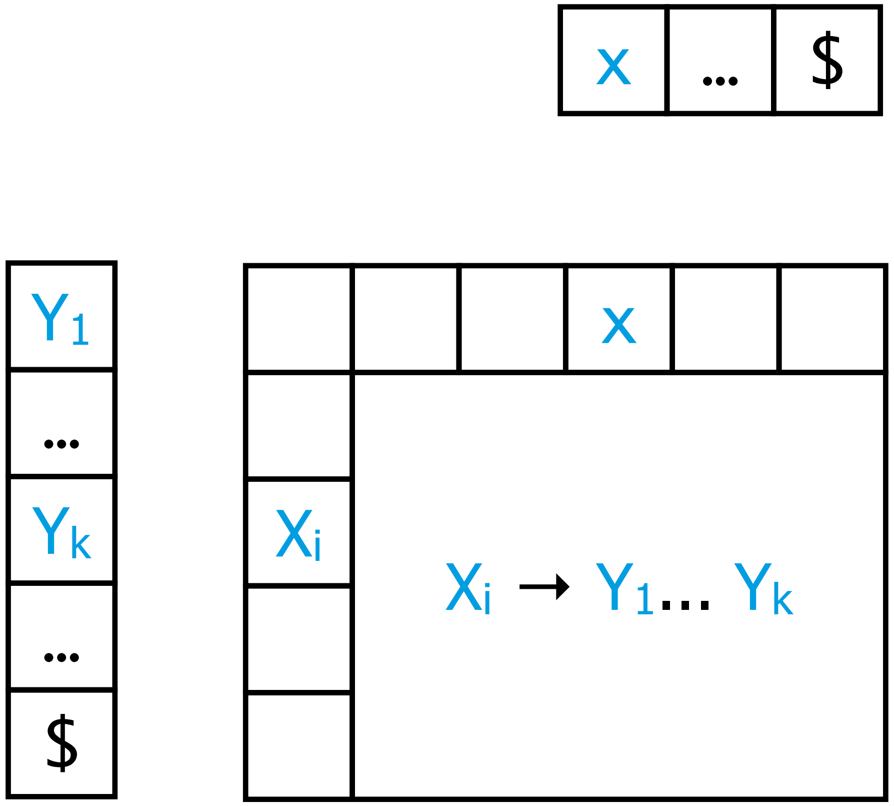
Predictive parsing

automaton



Predictive parsing

automaton



II

LL parse tables

Predictive parsing

filling the table

entry $(X, w) \in P$ at row X and column T

$T \in$ FIRST(w)

nullable(w) $\wedge$ $T \in$ FOLLOW(X)

Predictive parsing

filling the table

entry (X, w)∈P at row X and column T

T∈ $\boxed{\text{FIRST(w)}}$ — letters that w can start with

nullable(w) ∧ T∈ FOLLOW(X)

Predictive parsing
filling the table

entry $(X, w) \in P$ at row X and column T

$T \in$ FIRST(w)

$\boxed{\text{nullable}(w)} \wedge T \in$ FOLLOW(X)

$w \Rightarrow_G^* \varepsilon$

Predictive parsing

filling the table

entry (X, w)$\in$P at row X and column T

T$\in$ FIRST(w)

nullable(w) $\wedge$ T$\in$ $\boxed{\text{FOLLOW(X)}}$ letters that can follow X

Predictive parsing

nullable

nullable(X)

$(X, \varepsilon) \in P \Rightarrow$ nullable(X)

$(X_0, X_1 \ldots X_k) \in P \wedge$ nullable$(X_1) \wedge \ldots \wedge$ nullable$(X_k) \Rightarrow$ nullable(X_0)

nullable(w)

nullable(ε)

nullable$(X_1 \ldots X_k)$ = nullable$(X_1) \wedge \ldots \wedge$ nullable(X_k)

Predictive parsing

first sets

FIRST(X)

$X \in \Sigma$: FIRST(X) = {X}

$(X_0, X_1 \ldots X_i \ldots X_k) \in P \wedge \text{nullable}(X_1 \ldots X_i) \Rightarrow \text{FIRST}(X_0) \supseteq \text{FIRST}(X_{i+1})$

FIRST(w)

$\text{FIRST}(\varepsilon) = \{\}$

$\neg\text{nullable}(X) \Rightarrow \text{FIRST}(Xw) = \text{FIRST}(X)$

$\text{nullable}(X) \Rightarrow \text{FIRST}(Xw) = \text{FIRST}(X) \cup \text{FIRST}(w)$

Predictive parsing

follow sets

FOLLOW(X)

$(X_0, X_1 \ldots X_i \ldots X_k) \in P \wedge \text{nullable}(X_{i+1} \ldots X_k) \Rightarrow \text{FOLLOW}(X_i) \supseteq \text{FOLLOW}(X_0)$

$(X_0, X_1 \ldots X_i \ldots X_k) \in P \Rightarrow \text{FOLLOW}(X_i) \supseteq \text{FIRST}(X_{i+1} \ldots X_k)$

Example

p1: Exp → Term Exp′
p2: Exp′ → "+" Term Exp′
p3: Exp′ →
p4: Term → Fact Term′
p5: Term′ → "*" Fact Term′
p6: Term′ →
p7: Fact → Num
p8: Fact → "(" Exp ")"

	nullable	FIRST	FOLLOW
Exp			
Exp′			
Term			
Term′			
Fact			

Example
nullable

$(X, \varepsilon) \in P \Rightarrow \text{nullable}(X)$

$(X_0, X_1 \ldots X_k) \in P \wedge$

$\text{nullable}(X_1) \wedge \ldots \wedge \text{nullable}(X_k) \Rightarrow \text{nullable}(X_0)$

p1: Exp → Term Exp’
p2: Exp’ → “+” Term Exp’
p3: Exp’ →
p4: Term → Fact Term’
p5: Term’ → “*” Fact Term’
p6: Term’ →
p7: Fact → Num
p8: Fact → “(” Exp “)”

	nullable	FIRST	FOLLOW
Exp	no		
Exp’	yes		
Term	no		
Term’	yes		
Fact	no		

Example

FIRST sets

$(X_0, X_1 \ldots X_i \ldots X_k) \in P \wedge$

$\text{nullable}(X_1 \ldots X_i) \Rightarrow \text{FIRST}(X_0) \supseteq \text{FIRST}(X_{i+1})$

p1: Exp → Term Exp’
p2: Exp’ → “+” Term Exp’
p3: Exp’ →
p4: Term → Fact Term’
p5: Term’ → “*” Fact Term’
p6: Term’ →
p7: Fact → Num
p8: Fact → “(” Exp “)”

	nullable	FIRST	FOLLOW
Exp	no	Num (	
Exp’	yes	+	
Term	no	Num (	
Term’	yes	*	
Fact	no	Num (	

Example

FOLLOW sets

$(X_0, X_1 \ldots X_i \ldots X_k) \in P \wedge$

$\text{nullable}(X_{i+1} \ldots X_k) \Rightarrow \text{FOLLOW}(X_i) \supseteq \text{FOLLOW}(X_0)$

$(X_0, X_1 \ldots X_i \ldots X_k) \in P \Rightarrow \text{FOLLOW}(X_i) \supseteq \text{FIRST}(X_{i+1} \ldots X_k)$

p1: Exp → Term Exp’
p2: Exp’ → “+” Term Exp’
p3: Exp’ →
p4: Term → Fact Term’
p5: Term’ → “*” Fact Term’
p6: Term’ →
p7: Fact → Num
p8: Fact → “(” Exp “)”

	nullable	FIRST	FOLLOW
Exp	no	Num (	)
Exp’	yes	+	)
Term	no	Num (	+)
Term’	yes	*	+)
Fact	no	Num (	* +)

Example

LL parse table

entry (X, w)∈P at row X and column T

T∈ FIRST(w)

nullable(w) ∧ T∈ FOLLOW(X)

p1: Exp → Term Exp'
p2: Exp' → "+" Term Exp'
p3: Exp' →
p4: Term → Fact Term'
p5: Term' → "*" Fact Term'
p6: Term' →
p7: Fact → Num
p8: Fact → "(" Exp ")"

	+	*	Num	(	)
Exp			p1	p1	
Exp'	p2				p3
Term			p4	p4	
Term'	p6	p5			p6
Fact			p7	p8	

Example

parsing

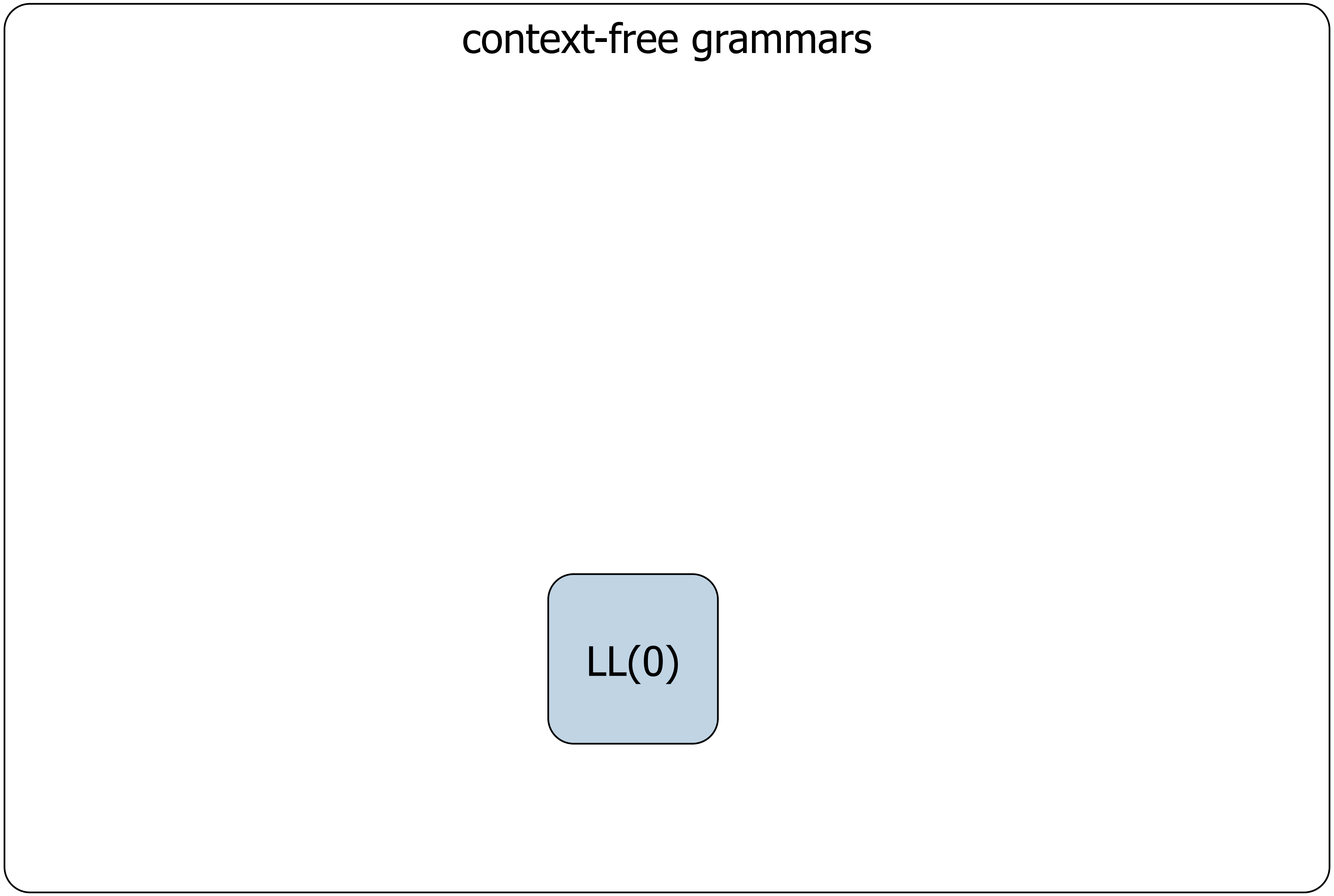
p1: Exp → Term Exp′
p2: Exp′ → "+" Term Exp′
p3: Exp′ →
p4: Term → Fact Term′
p5: Term′ → "*" Fact Term′
p6: Term′ →
p7: Fact → Num
p8: Fact → "(" Exp ")"

	+	*	Num	(	)
Exp			p1	p1	
Exp′	p2				p3
Term			p4	p4	
Term′	p6	p5			p6
Fact			p7	p8	

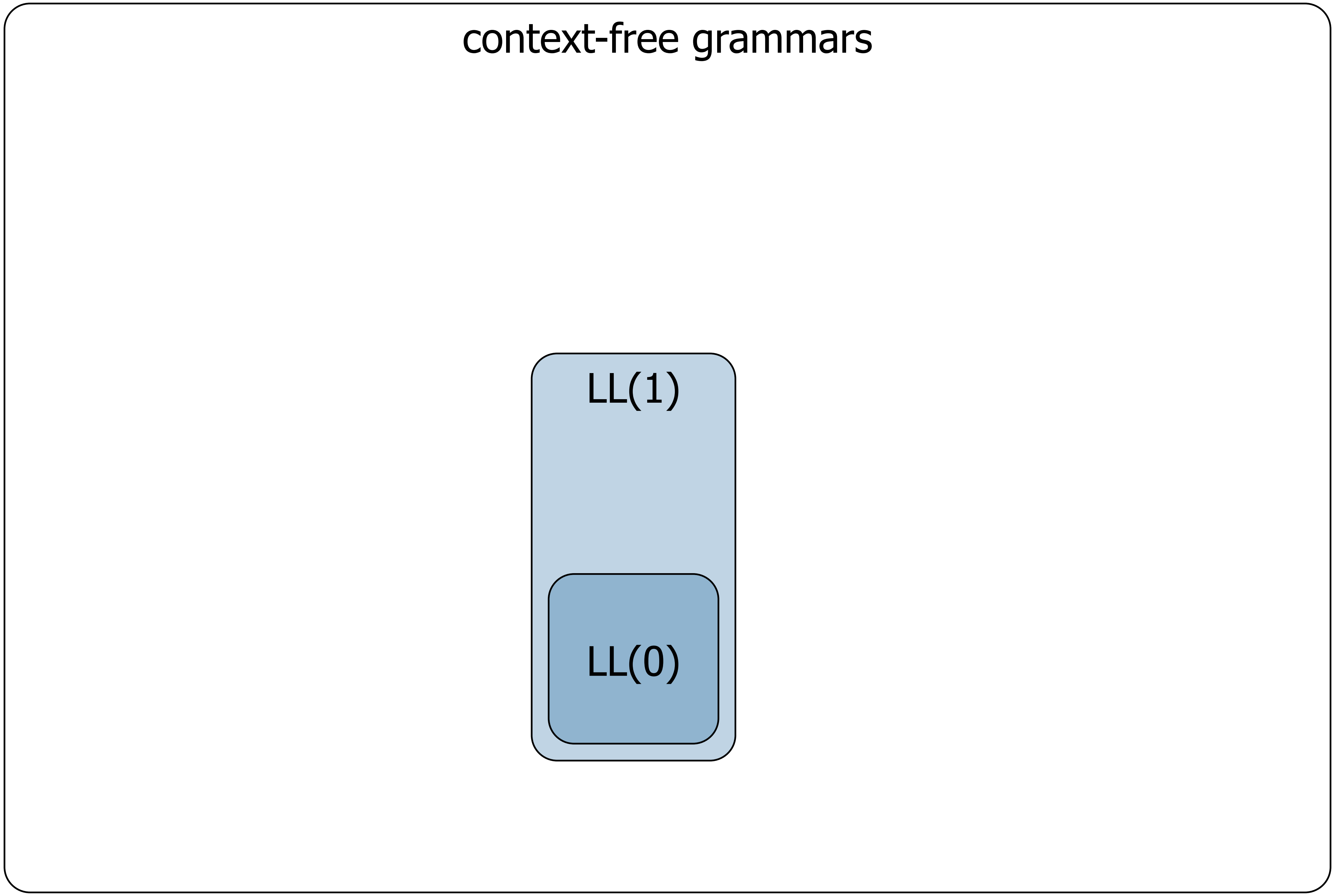
Grammar classes

context-free grammars

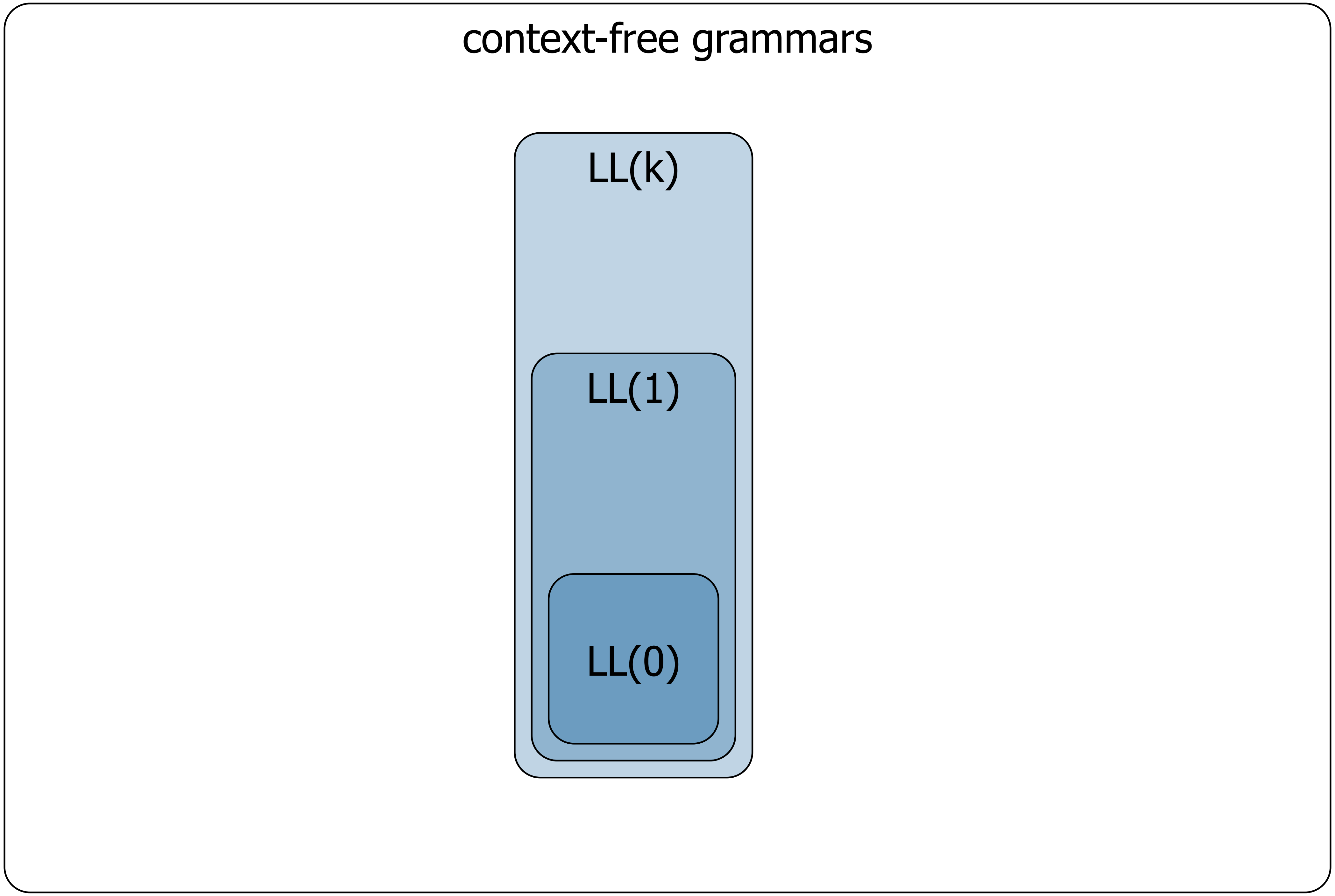
Grammar classes



Grammar classes



Grammar classes



Predictive parsing

encoding precedence

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

Fact → Num
Fact → "(" Exp ")"
Term → Term "*" Fact
Term → Fact
Exp → Exp "+" Term
Exp → Term

Predictive parsing
eliminating left recursion

Term → Term "*" Fact
Term → Fact
Exp → Exp "+" Term
Exp → Term

Term′ → "*" Fact Term′
Term′ →
Term → Fact Term′
Exp′ → "+" Term Exp′
Exp′ →
Exp → Term Exp′

Predictive parsing

left factoring

Exp → "if" Exp "then" Exp "else" Exp
Exp → "if" Exp "then" Exp

Exp → "if" Exp "then" Exp Else
Else → "else" Exp
Else →

TU Delft

III

summary

Summary

lessons learned

Summary

lessons learned

How can we parse context-free languages effectively?

- predictive parsing algorithms

Summary
lessons learned

How can we parse context-free languages effectively?

- predictive parsing algorithms

Which grammar classes are supported by these algorithms?

- LL(k) grammars, LL(k) languages

Summary

lessons learned

How can we parse context-free languages effectively?

- predictive parsing algorithms

Which grammar classes are supported by these algorithms?

- LL(k) grammars, LL(k) languages

How can we generate compiler tools from that?

- implement automaton
- generate parse tables

Summary
lessons learned

How can we parse context-free languages effectively?

- predictive parsing algorithms

Which grammar classes are supported by these algorithms?

- LL(k) grammars, LL(k) languages

How can we generate compiler tools from that?

- implement automaton
- generate parse tables

Literature

learn more

Literature

learn more

formal languages

Noam Chomsky: Three models for the description of language. 1956

J. E. Hopcroft, R. Motwani, J. D. Ullman: Introduction to Automata Theory, Languages, and Computation. 2006

Literature

learn more

formal languages

Noam Chomsky: Three models for the description of language. 1956

J. E. Hopcroft, R. Motwani, J. D. Ullman: Introduction to Automata Theory, Languages, and Computation. 2006

syntactical analysis

Andrew W. Appel, Jens Palsberg: Modern Compiler Implementation in Java, 2nd edition. 2002

Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Monica S. Lam: Compilers: Principles, Techniques, and Tools, 2nd edition. 2006

Outlook

coming next

lectures

- last lecture: LR parsing

Question & Answer Jan 10

- 10 questions, submit & vote

Lab Dec 15

- translate expressions & statements
- challenge: stack limits

copyrights

Pictures
copyrights

Slide 1: Book Scanner by Ben Woosley, some rights reserved

Slides 5-7: Noam Chomsky by Fellowsisters, some rights reserved

Slide 8, 9, 26-28: Tiger by Bernard Landgraf, some rights reserved

Slide 32: Ostsee by Mario Thiel, some rights reserved