Non-Classical Parsing

Special Course at the
Eötvös Loránd University

Henning Bordihn

Universität Potsdam
Institut für Informatik und Computational Science

Teaching at Universities

- Mathematics, mainly Algebra
- Theoretical Computer Science
- Logic
- Programming
- Software Engineering
- Compiler (and Program Transformation)
- several special courses, (automata, formal languages, information theory, ...)

- This course is financially supported by the Erasmus program of the EU.
- Contracts with the University of Potsdam exist.
- It is also open for students. You may spend a term or two in Potsdam.

Potsdam, the center of Prussian kings, is a direct neighbour of the city of Berlin.

Outline

1. Non-context-free phenomena
2. Non-context-free descriptors
3. Efficient parsing algorithms for non-context-free mechanisms (for CD grammar systems)
4. Summary

Outline

1. Non-context-free phenomena
2. Non-context-free descriptors
3. Efficient parsing algorithms for non-context-free mechanisms (for CD grammar systems)
4. Summary

Non-Context-Free Phenomena

- Programming languages
- Linguistics
- Developmental biology
- Molecular genetics
- Logic (*language of tautologies*)
- Economic modeling (*workflows*)
- ...

Non-Context-Free Phenomena

- Programming languages
- Linguistics
- Developmental biology
- Molecular genetics
- Logic (*language of tautologies*)
- Economic modeling (*workflows*)
- ...

Programming Languages: Static Semantic Constraints

- Consider the language of all executable Java programs L_{Java}.

- Assume L_{Java} be context-free.

Programming Languages: Static Semantic Constraints

- Consider the language of all executable Java programs L_{Java}.
- Assume L_{Java} be context-free.
- Further consider the following regular language R:

```
class A {
    int x(0|1)*;

    public static void main(String[] args) {
        System.out.println(x(0|1)*);
    }
}
```

Programming Languages: Static Semantic Constraints

- Known fact: $L_{\text{Java}} \cap R$ is context-free

Programming Languages: Static Semantic Constraints

- Known fact: $L_{\mathsf{Java}} \cap R$ is context-free

- Consider the homomorphism h with

$$h(0) = 0$$
$$h(1) = 1$$
$$h(\cdot) = \varepsilon$$

- Known fact: $h(L_{\mathsf{Java}} \cap R)$ is context-free

Programming Languages: Static Semantic Constraints

- Known fact: $L_{\text{Java}} \cap R$ is context-free

- Consider the homomorphism h with

$$h(0) = 0$$
$$h(1) = 1$$
$$h(\cdot) = \varepsilon$$

- Known fact: $h(L_{\text{Java}} \cap R)$ is context-free

- However, $h(L_{\text{Java}} \cap R) = \{ ww \mid w \in \{0,1\}^* \}$
  is not context-free, a contradiction.

Linguistics: Swiss German Word Order

- Jan säit das mer em Hans es huus hälfed aastriiche.
  John said that we Hans the house helped paint.
  John said that we helped Hans paint the house.

Linguistics: Swiss German Word Order

- Jan säit das mer em Hans es huus hälfed aastriiche.
  John said that we Hans the house helped paint.
  John said that we helped Hans paint the house.
- Jan säit das mer d'chind em Hans es huus lönd hälfe aastriiche.
  John said that we the children Hans the house let help paint.
  John said that we let the children help Hans paint the house.

Linguistics: Swiss German Word Order

- Jan säit das mer em Hans es huus hälfed aastriiche.
  John said that we Hans the house helped paint.
  John said that we helped Hans paint the house.

- Jan säit das mer d'chind em Hans es huus lönd hälfe aastriiche.
  John said that we the children Hans the house let help paint.
  John said that we let the children help Hans paint the house.

- Jan säit das mer $(\text{d'chind})^i$ $(\text{em Hans})^j$ es huus haend wele $(\text{laa})^i$ $(\text{hälfe})^j$ aastriiche.
  John said that we $(\text{the children})^i$ $(\text{Hans})^j$ the house have wanted to $(\text{let})^i$ $(\text{help})^j$ paint.
  John said that we wanted to let Mary help Hans, Frank help Jessica, Chris help Lucy, and Vanessa help René paint the house.

Linguistics: Swiss German Word Order

- Mapping:
  - accusative case objects (d'chind) to a,
  - dative case objects (Hans) to b,
  - verbs requiring accusative case (laa), to c,
  - verbs requiring dative case (hälfe), to d,
  - erase everything else,

  Result: $a^i b^j c^i d^j$ for all sentences of this form:

  $\{ a^i b^j c^i d^j \mid i \geq 1, j \geq 1 \}$

Linguistics: Swiss German Word Order

- Mapping:
  - accusative case objects (d'chind) to a,
  - dative case objects (Hans) to b,
  - verbs requiring accusative case (laa), to c,
  - verbs requiring dative case (hälfe), to d,
  - erase everything else,

  Result: $a^i b^j c^i d^j$ for all sentences of this form:

  $$\{ a^i b^j c^i d^j \mid i \geq 1,\ j \geq 1 \}$$

- If the verbs requiring accusative and dative case are mapped to a and b, respectively, then a subset of

  $\{ ww \mid w \in \{a, b\}^+,\ |w| \geq 2 \}$ is obtained.

Developmental Biology: Cell Division

- Growth/Development of organisms (cell division):

$$A \rightarrow BB$$

- Performed (almost) in *parallel*
- Sentential form $A \quad A \quad A \quad A$
  should rewrite to $BB\,BB\,BB\,BB$
- Non-context-freeness due to exponential growth

Frontiers of classical grammar models

- *What we have seen:*
  For several application areas, context-free grammars are not enough to model all relevant aspects.

Frontiers of classical grammar models

- *What we have seen:*
  For several application areas, context-free grammars are not enough to model all relevant aspects.

- *What can we do?*
  - Like in Compilers: exclude those aspects from the syntax spacification and use static semantics (rules of well-formedness instead).
  - Use more powerfull mechanisms.

Frontiers of classical grammar models

- *What we have seen:*
  For several application areas, context-free grammars are not enough to model all relevant aspects.

- *What can we do?*
  - Like in Compilers: exclude those aspects from the syntax spacification and use static semantics (rules of well-formedness instead).
  - Use more powerfull mechanisms.

- *Can we use type-1 or type-0 grammars?*
  In principle yes, but those mechanisms are not feasible.

Frontiers of classical grammar models

- *What we have seen:*
  For several application areas, context-free grammars are not enough to model all relevant aspects.

- *What can we do?*
  - Like in Compilers: exclude those aspects from the syntax spacification and use static semantics (rules of well-formedness instead).
  - Use more powerfull mechanisms.

- *Can we use type-1 or type-0 grammars?*
  In principle yes, but those mechanisms are not feasible.

  $\hookrightarrow$ **Different approaches?!**

Outline

1. Non-context-free phenomena
2. Non-context-free descriptors
3. Efficient parsing algorithms for non-context-free mechanisms (for CD grammar systems)
4. Summary

Adding to the power of context-free mechanisms

- Controlled derivations
  Ruling out derivations according to certain criteria

Adding to the power of context-free mechanisms

- Controlled derivations
  Ruling out derivations according to certain criteria

- Parallel derivations (Lindenmayer systems)
  Iterated finite substitutions/homomorphisms

Adding to the power of context-free mechanisms

- Controlled derivations
  Ruling out derivations according to certain criteria
- Parallel derivations (Lindenmayer systems)
  Iterated finite substitutions/homomorphisms
- Mildly context-sensitive mechanisms
  TAGs, Head-Grammars, combinatory categorial grammars etc.

Adding to the power of context-free mechanisms

- Controlled derivations
  Ruling out derivations according to certain criteria

- Parallel derivations (Lindenmayer systems)
  Iterated finite substitutions/homomorphisms

- Mildly context-sensitive mechanisms
  TAGs, Head-Grammars, combinatory categorial grammars etc.

- Grammar systems
  Cooperation of several context-free grammars

Controlled Derivations—Example

Matrix grammars [Ábrahám (1965)]

- Line up rules to finite sequences, e.g.:

  $(S \rightarrow AB),\ (A \rightarrow aAb,\ B \rightarrow cB),\ (A \rightarrow ab,\ B \rightarrow c)\ \rightarrow\ \{\, a^n b^n c^n \mid n \geq 1 \,\}$

Controlled Derivations—Example

Matrix grammars [Ábrahám (1965)]

- Line up rules to finite sequences, e.g.:

  $(S \to AB),\ (A \to aAb,\ B \to cB),\ (A \to ab,\ B \to c)\ \to\ \{\, a^n b^n c^n \mid n \geq 1 \,\}$

- **Appearance checking (ac):** set of occurrences of rules that can be left out if not applicable to the sentential form (*here: empty*)

Parallel Derivations—Lindenmayer systems

- Modeling of biological developmental processes [Lindenmayer 68], ...
- All symbols can be rewritten
- Parallel replacements of all symbols
- *Example:* $\{ a^{2^n} \mid n \geq 0 \}$ with $a \to a^2$

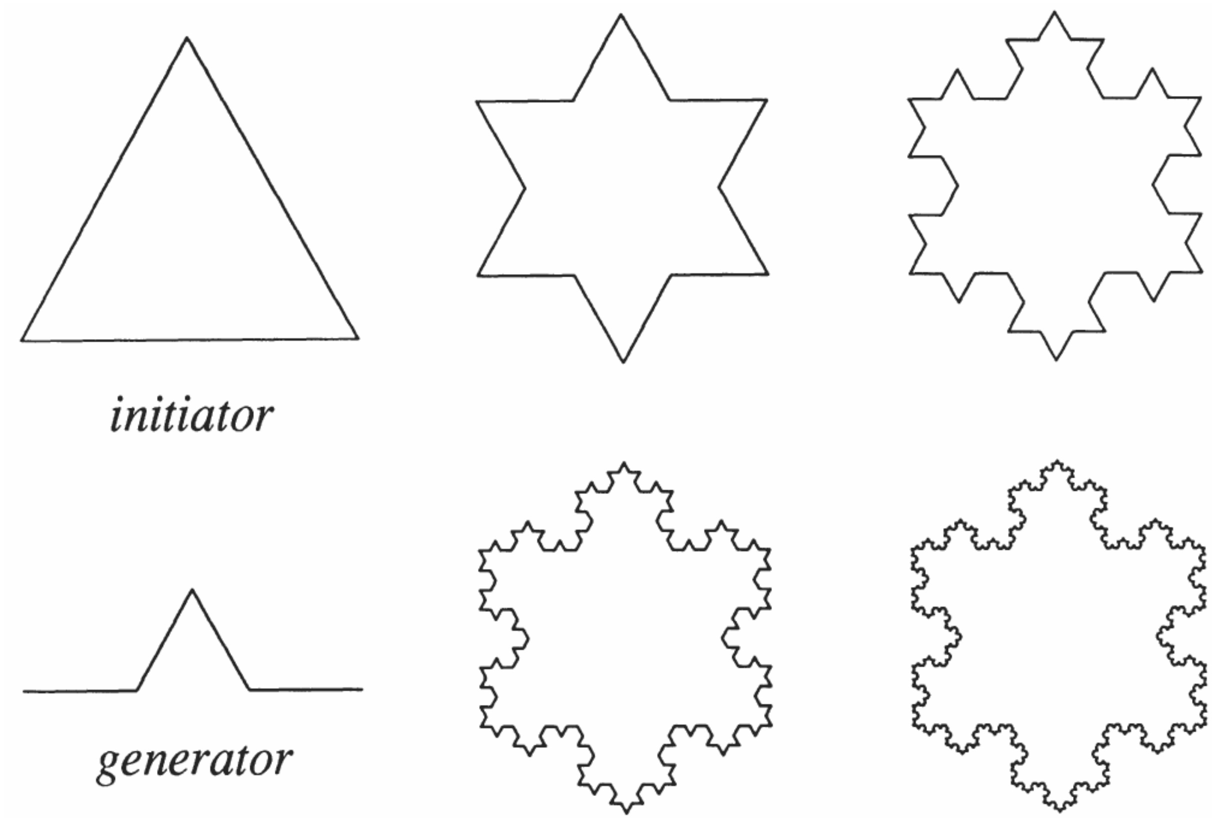
$$a \Longrightarrow aa \Longrightarrow aaaa \Longrightarrow a^8 \Longrightarrow a^{16} \Longrightarrow \ldots$$

Selected Applications

Reference:

Przemyslaw Prusinkiewicz, Aristid Lindenmayer,
The Algorithmic Beauty of Plants, Springer 1990.

Fractals

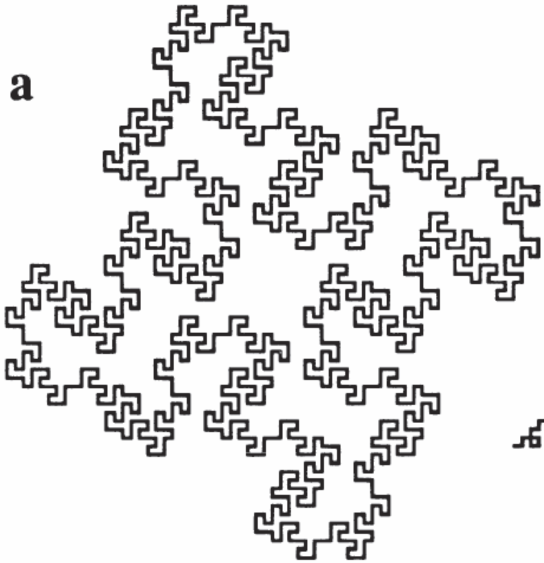


Turtle Graphic

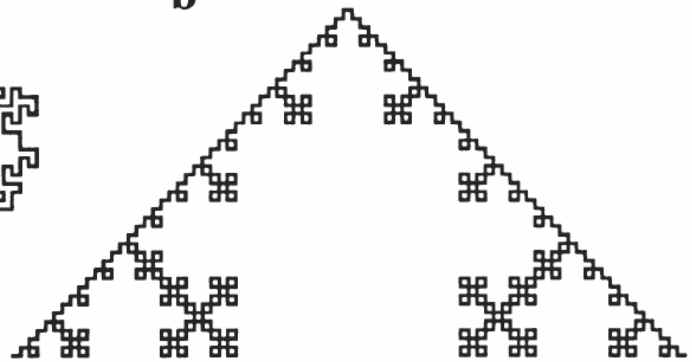
α, δ	two angles
α	initial angle with x-axis
F	draw line of unit length "'straightforward"'
+	change direction by $+\delta$
-	change direction by $-\delta$

Fractals (2)

a



b

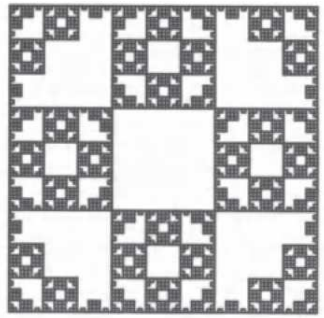


$n = 2,\ \delta = 90°$
F-F-F-F
F → F+FF-FF-F-F+F+F
F-F-F+F+FF+FF-F

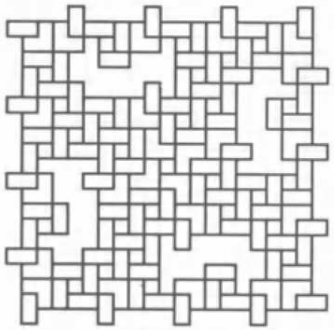
$n = 4,\ \delta = 90°$
-F
F → F+F-F-F+F

Fractals (3)

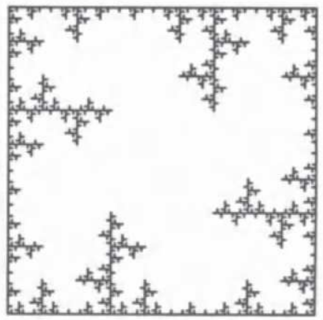
a n = 4, δ = 90°
F-F-F-F
F → FF-F-F-F-F-F+F



b n = 4, δ = 90°
F-F-F-F
F → FF-F-F-F-FF

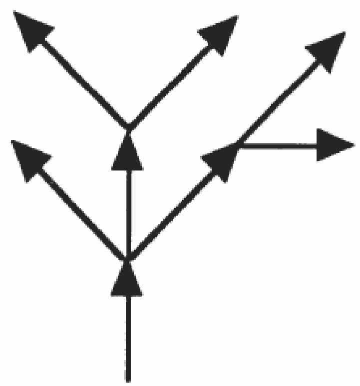


c n = 3, δ = 90°
F-F-F-F
F → FF-F+F-F-FF



d n = 4, δ = 90°
F-F-F-F
F → FF-F--F-F

Graphical Interpretation with Stack Operations



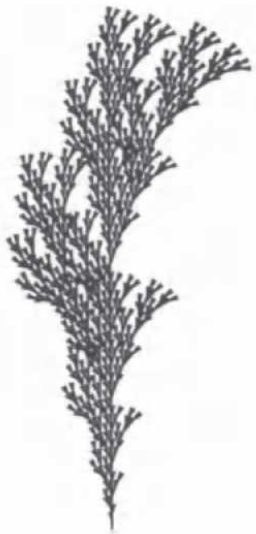
F[+F][-F[-F]F]F[+F][-F]

[push turtle state onto stack

] pop state from stack and set turtle to this state
(by moving turtle without drawing a line)

Branching Structures (1)

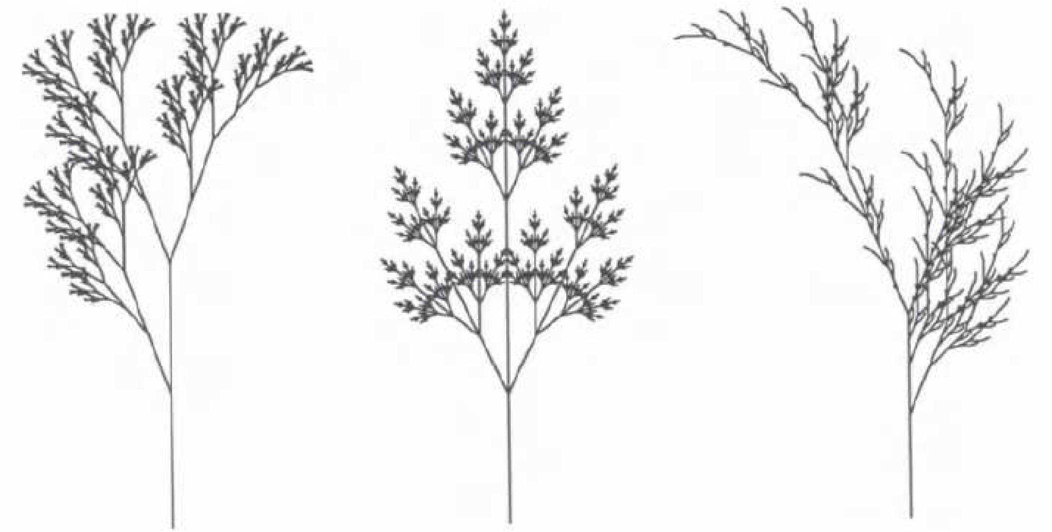
a
n=5, δ=25.7°
F
F $\rightarrow$ F[+F]F[-F]F



b
n=5, δ=20°
F
F $\rightarrow$ F[+F]F[-F][F]

c
n=4, δ=22.5°
F
F $\rightarrow$ FF-[-F+F+F]+
[+F-F-F]

Branching Structures (2)



d
n=7, δ=20°
X
X →F[+X]F[-X]+X
F →FF

e
n=7, δ=25.7°
X
X→F[+X][-X]FX
F→FF

f
n=5, δ=22.5°
X
X→F-[[X]+X]+F[+FX]-X
F→FF

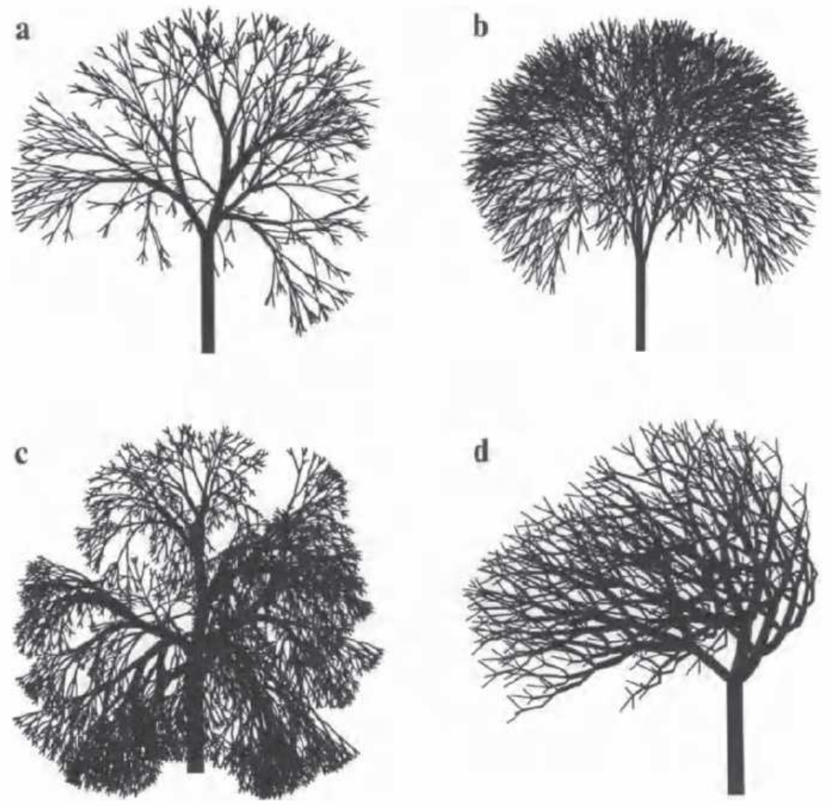
Three-Dimensional Graphics with Textures

n=7, δ=22.5°

ω : A
p_1 : A → [&FL!A]/////'[&FL!A]///////'[&FL!A]
p_2 : F → S ///// F
p_3 : S → F L
p_4 : L → ['''∧∧{-f+f+f-|-f+f+f}]

Three-Dimensional Graphics with Textures (2)

Parametrized Descriptions




```
#define d1 94.74       /* divergence angle 1 */
#define d2 132.63      /* divergence angle 2 */
#define a 18.95        /* branching angle */
#define lr 1.109       /* elongation rate */
#define vr 1.732       /* width increase rate */
```

ω : `!(1)F(200)/(45)A`

p_1 : `A : *` $\rightarrow$ `!(vr)F(50)[&(a)F(50)A]/(d1)`
`[&(a)F(50)A]/(d2)[&(a)F(50)A]`

p_2 : `F(l) : *` $\rightarrow$ `F(l*lr)`

p_3 : `!(w) : *` $\rightarrow$ `!(w*vr)`

Lindenmayer Systems: Variants

Determinism	exactly one rule per symbol $\hookrightarrow$ iterated homomorphisms
Tables	several rule sets
Extension	auxiliary symbols
Adult	ruling out all fixed points
...	

Lindenmayer Systems: Variants

Determinism	exactly one rule per symbol $\hookrightarrow$ iterated homomorphisms
Tables	several rule sets
Extension	auxiliary symbols
Adult	ruling out all fixed points
...	

ET0L systems — nondeterministic extended tabled L systems

0: applicability of rules depends on zero neighbouring symbols (context-free derivation)

Grammar Systems (GS)

- *Idea:* Several context-free grammars (*components*) jointly generate the strings of the language.

Grammar Systems (GS)

- *Idea:* Several context-free grammars (*components*) jointly generate the strings of the language.

- CD-GS: *Sequential cooperation* [Csuhaj-Várju, Dassow (1992)] working on a common sentential form in turns
  - Distributed problem solving in blackboard architectures
  - Multi-level grammars [Meersman, Rozenberg (1978)]
  - Sequential analogue to tabled Lindenmayer systems [Bordihn, Csuhaj-Varjú, Dassow (1997)]

Grammar Systems (GS)

- *Idea:* Several context-free grammars (*components*) jointly generate the strings of the language.

- CD-GS: *Sequential cooperation* [Csuhaj-Várju, Dassow (1992)] working on a common sentential form in turns
  - Distributed problem solving in blackboard architectures
  - Multi-level grammars [Meersman, Rozenberg (1978)]
  - Sequential analogue to tabled Lindenmayer systems [Bordihn, Csuhaj-Varjú, Dassow (1997)]

- PC-GS: *Parallel cooperation* [Păun, Sântean (1989)] autonomous, synchronized derivations and communication of sentential forms upon request (by particular nonterminal symbols)

CD Grammar Systems—Definition

- A context-free CDGS of degree n is a tuple

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n)$$

  - N is a finite set of nonterminal symbols,
  - T is a finite set of terminal symbols
  - $S \in N$ (axiom),
  - P_i $(1 \leq i \leq n)$ is a finite set of context-free productions of the form $A \to \alpha$, $A \in N$, $\alpha \in (N \cup T)^*$

$\hookrightarrow$ Each (N, T, S, P_i) $(1 \leq i \leq n)$ is a context-free grammar (component).

CD Grammar Systems—Definition

- A context-free CDGS of degree n is a tuple

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n)$$

  - N is a finite set of nonterminal symbols,
  - T is a finite set of terminal symbols
  - $S \in N$ (axiom),
  - P_i $(1 \leq i \leq n)$ is a finite set of context-free productions of the form $A \rightarrow \alpha$, $A \in N$, $\alpha \in (N \cup T)^*$

  $\hookrightarrow$ Each (N, T, S, P_i) $(1 \leq i \leq n)$ is a context-free grammar (component).

- $x \Longrightarrow_i y$ iff $\quad x = \gamma_1 A \gamma_2, \quad y = \gamma_1 \alpha \gamma_2, \quad A \rightarrow \alpha \in P_i$

Cooperation Strategies

- Components work sequentially (in turns) on a common sentential form
- are activated in a nondeterministic way

Cooperation Strategies

- Components work sequentially (in turns) on a common sentential form
- are activated in a nondeterministic way

Derivation mode	Number of steps to be performed (of a component, once activated)
$*$-mode	arbitrary
$=m$-mode	exactly m
$\leq m$-mode	at most m
$\geq m$-mode	at least m
t-mode	as many as possible
full-mode	until a nonterminal has been introduced that the component cannot replace

- $x \overset{=m}{\Longrightarrow}_i y$ iff $x = x_0 \Longrightarrow_i x_1 \Longrightarrow_i \cdots \Longrightarrow_i x_m = y$

- $x \overset{\mathrm{t}}{\Longrightarrow}_i y$ iff $x \overset{*}{\Longrightarrow}_i y$ and there is no z such that $y \Longrightarrow_i z$

- $x \stackrel{=m}{\Longrightarrow}_i y$ iff $x = x_0 \Longrightarrow_i x_1 \Longrightarrow_i \cdots \Longrightarrow_i x_m = y$

- $x \stackrel{\mathrm{t}}{\Longrightarrow}_i y$ iff $x \stackrel{*}{\Longrightarrow}_i y$ and there is no z such that $y \Longrightarrow_i z$

- For $\mu \in \{ =m, \mathrm{t} \mid m \geq 1 \}$:

  $$L(\Gamma, \mu) = \{ w \in T^* \mid S \stackrel{\mu}{\Longrightarrow}_{i_1} v_1 \stackrel{\mu}{\Longrightarrow}_{i_2} \ldots \stackrel{\mu}{\Longrightarrow}_{i_\ell} w, \ \ell \geq 1,\ 1 \leq i_j \leq n \text{ for } 1 \leq j \leq \ell \}$$

- $x \overset{=m}{\Longrightarrow}_i y$ iff $x = x_0 \Longrightarrow_i x_1 \Longrightarrow_i \cdots \Longrightarrow_i x_m = y$

- $x \overset{\mathrm{t}}{\Longrightarrow}_i y$ iff $x \overset{*}{\Longrightarrow}_i y$ and there is no z such that $y \Longrightarrow_i z$

- For $\mu \in \{ =m, \mathrm{t} \mid m \geq 1 \}$:

$$L(\Gamma, \mu) = \{ w \in T^* \mid S \overset{\mu}{\Longrightarrow}_{i_1} v_1 \overset{\mu}{\Longrightarrow}_{i_2} \ldots \overset{\mu}{\Longrightarrow}_{i_\ell} w, \\ \ell \geq 1,\ 1 \leq i_j \leq n \text{ for } 1 \leq j \leq \ell \}$$

Example:

$P_1 = \{S \to S,\ S \to AB\}$

$P_2 = \{A \to aA',\ B \to cB'\}$ $P_3 = \{A \to A'b,\ B \to B'd\}$

$P_4 = \{A' \to A,\ B' \to B\}$ $P_5 = \{A \to \lambda,\ B \to \lambda\}$

$$L(\Gamma, =2) = L(\Gamma, \mathrm{t}) = \{ a^i b^j c^i d^j \mid i, j \geq 0 \}$$

Outline

1. Non-context-free phenomena
2. Non-context-free descriptors
3. Efficient parsing algorithms for non-context-free mechanisms (for CD grammar systems)
4. Summary

The Goal

- **Restricting** CDGS (in t- and $=m$-modes) such that
  - efficient top-down parsing becomes possible
    $\hookrightarrow$ deterministic one-way parsing without backtracking (*here:* top-down),
  - important non-context-free languages can be generated

The Goal

- **Restricting** CDGS (in t- and $=m$-modes) such that
  - efficient top-down parsing becomes possible
    $\hookrightarrow$ deterministic one-way parsing without backtracking (*here:* top-down),
  - important non-context-free languages can be generated

$*$ Removing nondeterminism:

- Leftmost derivations
- $\mathrm{LL}(k)$-condition

Leftmost Derivations

- strong leftmost mode ($x \overset{\mu}{\underset{\mathrm{s}}{\Longrightarrow}}_i y$):
  always replace the leftmost nonterminal in the sentential form

  $\hookrightarrow$ leads to the family of context-free languages (in any derivation mode)

Leftmost Derivations

- strong leftmost mode ($x \overset{\mu}{\underset{s}{\Longrightarrow}}_i y$):
  always replace the leftmost nonterminal in the sentential form
  $\hookrightarrow$ leads to the family of context-free languages (in any derivation mode)

- Domain of component P_i, $1 \leq i \leq n$:

$$\mathrm{dom}(P_i) = \{ A \mid A \rightarrow \alpha \in P_i \text{ for some } \alpha \}$$

Leftmost Derivations

- strong leftmost mode ($x \overset{\mu}{\underset{\mathrm{s}}{\Longrightarrow}}_i y$):
  always replace the leftmost nonterminal in the sentential form
  $\hookrightarrow$ leads to the family of context-free languages (in any derivation mode)

- Domain of component P_i, $1 \leq i \leq n$:

$$\mathrm{dom}(P_i) = \{ A \mid A \rightarrow \alpha \in P_i \text{ for some } \alpha \}$$

- weak leftmost mode ($x \overset{\mu}{\underset{\mathrm{w}}{\Longrightarrow}}_i y$):
  - When no component has been activated yet (in the first step of some P_i): always replace the leftmost nonterminal in the sentential form
  - If the component P_i is already active: always replace the leftmost symbol in the sentential form that is from the domain $\mathrm{dom}(P_i)$

LL(k) condition for CDGS

Given: 1) CDGS Γ, $\gamma \in \{\mathrm{s}, \mathrm{w}\}$, $\mu \in \{\,\mathrm{t}, =m \mid m \geq 1\,\}$

2) Input to be analyzed $w = a_1 a_2 \ldots a_s \in T^*$

Question: Does $w \in L_\gamma(\Gamma, \mu)$ hold?

Goal: Re-construction of a leftmost derivation leading from S to w if $w \in L_\gamma(\Gamma, \mu)$

$\mathbf{LL}(k)$ condition for CDGS

Given: 1) CDGS Γ, $\gamma \in \{\mathrm{s}, \mathrm{w}\}$, $\mu \in \{\, \mathrm{t}, =m \mid m \geq 1 \,\}$
2) Input to be analyzed $w = a_1 a_2 \ldots a_s \in T^*$

Question: Does $w \in L_\gamma(\Gamma, \mu)$ hold?

Goal: Re-construction of a leftmost derivation leading from S to w if $w \in L_\gamma(\Gamma, \mu)$

Let $S \overset{\mu}{\underset{\gamma}{\Longrightarrow}}_{i_1} \ldots \overset{\mu}{\underset{\gamma}{\Longrightarrow}}_{i_j} a_1 a_2 \ldots a_r A y$ be already analyzed ($y \in V^*$).

LL(k) condition for CDGS

Given: 1) CDGS Γ, $\gamma \in \{\mathrm{s}, \mathrm{w}\}$, $\mu \in \{\, \mathrm{t}, =m \mid m \geq 1 \,\}$

2) Input to be analyzed $w = a_1 a_2 \dots a_s \in T^*$

Question: Does $w \in L_\gamma(\Gamma, \mu)$ hold?

Goal: Re-construction of a leftmost derivation leading from S to w if $w \in L_\gamma(\Gamma, \mu)$

Let $S \overset{\mu}{\underset{\gamma}{\Longrightarrow}}_{i_1} \dots \overset{\mu}{\underset{\gamma}{\Longrightarrow}}_{i_j} a_1 a_2 \dots a_r A y$ be already analyzed ($y \in V^*$).

LL(k) condition: Then the next k input symbols (tokens)
$a_{r+1} \dots a_{r+k}$ (look-ahead)
determine the unique next derivation step $\overset{\mu}{\underset{\gamma}{\Longrightarrow}}_{i_{j+1}}$

Context-Free $\mathrm{LL}(k)$ Parsing

Is using look-ahead $k > 1$ an issue?

- $\mathrm{LL}(1) \subset \mathrm{LL}(2) \subset \mathrm{LL}(3) \subset \ldots$

Context-Free LL(k) Parsing

Is using look-ahead $k > 1$ an issue?

- $\mathrm{LL}(1) \subset \mathrm{LL}(2) \subset \mathrm{LL}(3) \subset \ldots$
- Look, for example, at the following grammar:

$$Var \to SimpleVar \mid IndexedVar$$
$$SimpleVar \to \mathrm{Idf}$$
$$IndexedVar \to \mathrm{Idf}[Expr]$$

Context-Free LL(k) Parsing

Is using look-ahead $k > 1$ an issue?

- $\mathrm{LL}(1) \subset \mathrm{LL}(2) \subset \mathrm{LL}(3) \subset \ldots$
- Look, for example, at the following grammar:

$$\begin{aligned}
&Var \rightarrow SimpleVar \mid IndexedVar\\
&SimpleVar \rightarrow \mathrm{Idf}\\
&IndexedVar \rightarrow \mathrm{Idf}[Expr]
\end{aligned}$$

- Var-rule requires look-ahead of size 2
- Can hardly be improved since $SimpleVar$ is likely to be used in many other rules

$\mathbf{LL}(k)$ Parsing Tables

- Like $\mathrm{LL}(1)$ parsing tables,
  but provide a column for any token string of length $\leq k$

$\mathrm{LL}(k)$ Parsing Tables

- Like $\mathrm{LL}(1)$ parsing tables,
  but provide a column for any token string of length $\leq k$
- Generalize First and Follow sets to $\mathrm{FIRST}_k(X)$ and $\mathrm{FOLLOW}_k(X)$

$\mathbf{LL}(k)$ Parsing Tables

- Like $\mathrm{LL}(1)$ parsing tables,
  but provide a column for any token string of length $\leq k$
- Generalize First and Follow sets to $\mathrm{FIRST}_k(X)$ and $\mathrm{FOLLOW}_k(X)$
- *Naive approach:* For each production $A \rightarrow \alpha$ do:
  For each $w \in \mathrm{FIRST}_k(\mathrm{FIRST}_k(\alpha)\mathrm{FOLLOW}_k(A))$, set
  $T[A, w] = A \rightarrow \alpha$.

$\mathrm{LL}(k)$ Parsing Tables

- Like $\mathrm{LL}(1)$ parsing tables, but provide a column for any token string of length $\leq k$
- Generalize First and Follow sets to $\mathrm{FIRST}_k(X)$ and $\mathrm{FOLLOW}_k(X)$
- *Naive approach:* For each production $A \to \alpha$ do: For each $w \in \mathrm{FIRST}_k(\mathrm{FIRST}_k(\alpha)\mathrm{FOLLOW}_k(A))$, set $T[A, w] = A \to \alpha$.
- Unfortunately, this works only for **strong** $\mathrm{LL}(k)$ grammars.
- Every $\mathrm{LL}(1)$ grammar is strong $\mathrm{LL}(1)$, but strongness is a restriction for $\mathrm{LL}(k)$ grammars if $k > 1$.

LL(2) *versus* Strong LL(2)

- CFG

$$S \to aAaa \mid bAba$$
$$A \to b \mid \varepsilon$$

results in $\{aaa, abaa, bba, bbba\}$

LL(2) *versus* Strong LL(2)

- CFG

$$S \rightarrow aAaa \mid bAba$$
$$A \rightarrow b \mid \varepsilon$$

  results in $\{aaa, abaa, bba, bbba\}$

- Intuitively $\mathrm{LL}(2)$:
  two symbols of look-ahead are enough to predict the production

LL(2) *versus* Strong LL(2)

- CFG

$$S \to aAaa \mid bAba$$
$$A \to b \mid \varepsilon$$

  results in $\{aaa,\ abaa,\ bba,\ bbba\}$

- Intuitively $\mathrm{LL}(2)$:
  two symbols of look-ahead are enough to predict the production

- Is **not strong** $\mathrm{LL}(2)$ since
  $T[A, ba] = \{b, \varepsilon\}$.

LL(2) *versus* Strong LL(2)

- CFG

$$S \to aAaa \mid bAba$$
$$A \to b \mid \varepsilon$$

  results in $\{aaa,\ abaa,\ bba,\ bbba\}$

- Intuitively $\mathrm{LL}(2)$:
  two symbols of look-ahead are enough to predict the production

- Is **not strong** $\mathrm{LL}(2)$ since
  $T[A, ba] = \{b, \varepsilon\}$.

- Construction of strong $\mathrm{LL}(2)$ tables disregards "history" of derivation

- Can be fixed by using distinct copies of A_1 and A_2 of A

Parsing Tables for CDGS

- Every element in a parsing table must determine
  - the next component to be activated (its label) and
  - the sequence of productions to be applied while this component is active

Parsing Tables for CDGS

- Every element in a parsing table must determine
  - the next component to be activated (its label) and
  - the sequence of productions to be applied while this component is active

- **Simplification:** only the label of the component is needed,
  if the CDGS is deterministic, i.e., each P_i contains at most one rule for every nonterminal symbol.

Example

$$P_1 = \{S \to S,\ S \to AB\}$$
$$P_2 = \{A \to aA',\ B \to cB'\} \qquad P_3 = \{A \to A'b,\ B \to B'd\}$$
$$P_4 = \{A' \to A,\ B' \to B\} \qquad P_5 = \{A \to \lambda,\ B \to \lambda\}$$

$$L_{\mathrm{w}}(\Gamma, =2) = L_{\mathrm{w}}(\Gamma, \mathrm{t}) = \{\, a^i b^j c^i d^j \mid i, j \geq 0 \,\}$$

	a	b	c	d	λ
S	1	1	–	–	1
A	2	3	5	5	5
A'	4	4	4	4	4

Example

$$P_1 = \{S \to S,\ S \to AB\}$$
$$P_2 = \{A \to aA',\ B \to cB'\} \qquad P_3 = \{A \to A'b,\ B \to B'd\}$$
$$P_4 = \{A' \to A,\ B' \to B\} \qquad P_5 = \{A \to \lambda,\ B \to \lambda\}$$

$$L_{\mathrm{w}}(\Gamma, =2) = L_{\mathrm{w}}(\Gamma, \mathrm{t}) = \{\, a^i b^j c^i d^j \mid i, j \geq 0 \,\}$$

	a	b	c	d	λ
S	1	1	–	–	1
A	2	3	5	5	5
A'	4	4	4	4	4

$\hookrightarrow$ satisfies the condition for $\mathrm{LL}(1)$ CDGS

Computational Power of $\mathrm{LL}(k)$ CDGS

[Bordihn, Vaszil 2007]

- One can describe:
  - all context-free $\mathrm{LL}(k)$ languages (in $=1$-mode)

Computational Power of $\mathrm{LL}(k)$ CDGS

[Bordihn, Vaszil 2007]

- One can describe:
  - all context-free $\mathrm{LL}(k)$ languages (in $=1$-mode)
  - $\{\, a^i b^j c^i d^j \mid i,j \geq 0 \,\}$, $\{\, wcw \mid w \in \{a,b\}^* \,\}$, $\{\, a^i b^i c^i \mid i \geq 0 \,\}$ (using weak leftmost derivations)
  - languages that are not semi-linear (using weak leftmost derivations)

Parsing Algorithm—Idea

- **Weak leftmost derivations:** symbols have to be replaced that may appear far away from the leftmost nonterminal.

$$S \overset{=2}{\underset{\mathrm{w}}{\Longrightarrow}}_1 AB \overset{=2}{\underset{\mathrm{w}}{\Longrightarrow}}_2 aAcB \overset{=2}{\underset{\mathrm{w}}{\Longrightarrow}}_2 \ldots \overset{=2}{\underset{\mathrm{w}}{\Longrightarrow}}_2 a^r Ac^r B \ldots$$

Parsing Algorithm—Idea

- **Weak leftmost derivations:** symbols have to be replaced that may appear far away from the leftmost nonterminal.

$$S \underset{\mathrm{w}}{\overset{=2}{\Longrightarrow}}_1 AB \underset{\mathrm{w}}{\overset{=2}{\Longrightarrow}}_2 aAcB \underset{\mathrm{w}}{\overset{=2}{\Longrightarrow}}_2 \dots \underset{\mathrm{w}}{\overset{=2}{\Longrightarrow}}_2 a^r A c^r B \dots$$

- Add queues containing "pending productions" (together with the number of the corresponding derivation step) to a balanced binary search tree

Parsing Algorithm—Idea

- **Weak leftmost derivations:** symbols have to be replaced that may appear far away from the leftmost nonterminal.

$$S \xRightarrow[\text{w}]{=2}_1 AB \xRightarrow[\text{w}]{=2}_2 aAcB \xRightarrow[\text{w}]{=2}_2 \ldots \xRightarrow[\text{w}]{=2}_2 a^r Ac^r B \ldots$$

- Add queues containing "pending productions" (together with the number of the corresponding derivation step) to a balanced binary search tree
- Remove productions from those queues when they are used later and delete empty queues from the search tree

Parsing Algorithm—Idea

- **Weak leftmost derivations:** symbols have to be replaced that may appear far away from the leftmost nonterminal.

$$S \underset{\mathrm{w}}{\overset{=2}{\Longrightarrow}}_1 AB \underset{\mathrm{w}}{\overset{=2}{\Longrightarrow}}_2 aAcB \underset{\mathrm{w}}{\overset{=2}{\Longrightarrow}}_2 \ldots \underset{\mathrm{w}}{\overset{=2}{\Longrightarrow}}_2 a^r Ac^r B \ldots$$

- Add queues containing "pending productions" (together with the number of the corresponding derivation step) to a balanced binary search tree
- Remove productions from those queues when they are used later and delete empty queues from the search tree
- Control usage of productions stored in queues with the help of stacks that, for every occurrence of a nonterminal in the sentential form, store the time (derivation step) when it has been generated.

Parsing Algorithm—Result

Theorem. Let Γ be a CDGS working in $=m$-mode $(m \geq 2)$ with weak leftmost derivaions, and let Γ satisfy the $\mathrm{LL}(k)$ condition for CDGS $(k \geq 1)$.

Given the parsing table for Γ, one can effectively construct a parser for $L_{\mathrm{w}}(\Gamma, =m)$.

For every input string, the parser terminates in $O(n \cdot \log^2 n)$ time, where n is the length of the input.

When the Parsing Table can be Constructed

- The parsing table for CDGS can be constructed if it is strong $\mathrm{LL}(k)$.

When the Parsing Table can be Constructed

- The parsing table for CDGS can be constructed if it is strong $\mathrm{LL}(k)$.

- A CDGS $\Gamma = (V, \Sigma, S, P_1, \ldots, P_n)$ is strong-$\mathrm{LL}(k)$ if:

  1. $A \to \alpha \in P_i$ implies $A \to \alpha \notin P_j$ for all $j \neq i$
  2. The CFG $(V \setminus \Sigma, \Sigma, S, \bigcup_{1 \leq i \leq n} P_i)$ is strong $\mathrm{LL}(k)$.

When the Parsing Table can be Constructed

- The parsing table for CDGS can be constructed if it is strong $\mathrm{LL}(k)$.
- A CDGS $\Gamma = (V, \Sigma, S, P_1, \ldots, P_n)$ is strong-$\mathrm{LL}(k)$ if:

  1. $A \to \alpha \in P_i$ implies $A \to \alpha \notin P_j$ for all $j \neq i$
  2. The CFG $(V \setminus \Sigma, \Sigma, S, \bigcup_{1 \leq i \leq n} P_i)$ is strong $\mathrm{LL}(k)$.

- All relevant CDGS are strong $\mathrm{LL}(k)$.

Outline

1. Non-context-free phenomena
2. Non-context-free descriptors
3. Accepting grammars
4. Efficient parsing algorithms for non-context-free mechanisms (for CD grammar systems)
5. Summary

Summary

- $\mathrm{LL}(k)$ CDGS describe all "classical" context-free $\mathrm{LL}(k)$ languages as well as crucial non-context-free languages.
- $\mathrm{LL}(k)$ CDGS have an efficient parser (with $O(n \cdot \log^2 n)$ time complexity in the worst case).
- The $\mathrm{LL}(k)$-hierarchy collapses for CDGS at the first level. (Condition: Parsing table is given!)
- If a CDGS is strong-$\mathrm{LL}(k)$, then its parsing table can effectively be constructed. All relevant examples are strong-$\mathrm{LL}(k)$.