On Nondeterminism in Programmed Grammars

Alexander Meduna, Lukáš Vrábel, and Petr Zemek

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 00 Brno, Czech Republic
http://www.fit.vutbr.cz/∼{meduna, ivrabel, izemek}

13th International Conference on Automata and Formal Languages, 2011

Example

$$
\begin{aligned}
&(1\colon S \to ABC, \{2,5\})\\
&(2\colon A \to aA, \{3\})\\
&(3\colon B \to bB, \{4\})\\
&(4\colon C \to cC, \{2,5\})\\
&(5\colon A \to a, \{6\})\\
&(6\colon B \to b, \{7\})\\
&(7\colon C \to c, \emptyset)
\end{aligned}
$$

$$
\begin{aligned}
(S, 1) &\Rightarrow (ABC, 2)\\
&\Rightarrow (aABC, 3)\\
&\Rightarrow (aAbBC, 4)\\
&\Rightarrow (aAbBcC, 5)\\
&\Rightarrow (aabBcC, 6)\\
&\Rightarrow (aabbcC, 7)\\
&\Rightarrow (aabbcc, \bot)
\end{aligned}
$$

$$
\begin{tikzcd}
 & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[ll, bend right] \arrow[ddll] \\
1 \arrow[ur] \arrow[dr] & & & \\
 & 5 \arrow[r] & 6 \arrow[r] & 7
\end{tikzcd}
$$

$$L(G) = \{a^n b^n c^n \mid n \geq 1\}$$

Three types of nondeterminism:

1. Which rule should be chosen as the first one?
2. Which occurrence of a nonterminal should be rewritten?
3. Which successor should be chosen?

Three types of nondeterminism:

1. Which rule should be chosen as the first one?
2. Which occurrence of a nonterminal should be rewritten?
3. Which successor should be chosen?

What has been studied:

- at most one successor [Bordihn, Holzer 2006]
- at most two successors [Bordihn, Holzer 2006]
- graphs from various classes [BBDDFILLN, 2006]

Central topic of our paper:

- continue the study of the role of nondeterminism in programmed grammars

Motivation:

- theoretical: normal forms
- practical: parsing

Results: Number of Nondeterministic Rules

Definition

Nondeterministic rule: a rule with more than one successor.

Example

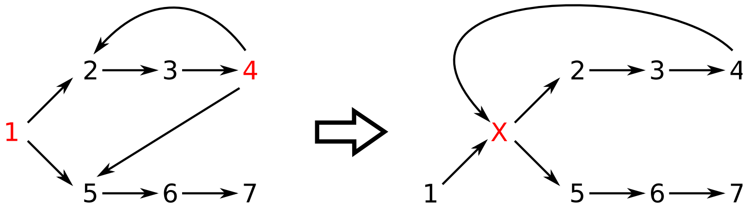
Nondeterministic rule: $(1\colon S \rightarrow ABC, \{2, 5\})$
Deterministic rule: $(2\colon A \rightarrow aA, \{3\})$

Definition

Nondeterministic rule: a rule with more than one successor.

Result

Every programmed grammar can be transformed into an equivalent programmed grammar with only a single nondeterministic rule.



Definition

Nondeterministic rule: a rule with more than one successor.

Proof Idea

For the original rule $(1: S \rightarrow ABC, \{2, 5\})$, we introduce the following rules:

Definition

Nondeterministic rule: a rule with more than one successor.

Proof Idea

For the original rule $(1 \colon S \rightarrow ABC, \{2, 5\})$, we introduce the following rules:

- $(1 \colon S \rightarrow \langle 1 \rangle \$, \{X\})$

Definition

Nondeterministic rule: a rule with more than one successor.

Proof Idea

For the original rule $(1\colon S \to ABC, \{2, 5\})$, we introduce the following rules:

- $(1\colon S \to \langle 1 \rangle \$, \{X\})$
- $(X\colon \$ \to \varepsilon, \{\dots, r, s, \dots\})$

Definition

Nondeterministic rule: a rule with more than one successor.

Proof Idea

For the original rule $(1\colon S \rightarrow ABC, \{2, 5\})$, we introduce the following rules:

- $(1\colon S \rightarrow \langle 1 \rangle \$, \{X\})$
- $(X\colon \$ \rightarrow \varepsilon, \{\dots, r, s, \dots\})$
- $(r\colon \langle 1 \rangle \rightarrow ABC, \{2\})$

Definition

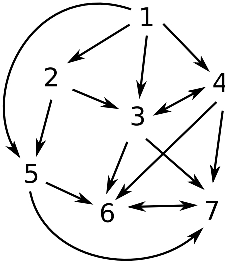
Nondeterministic rule: a rule with more than one successor.

Proof Idea

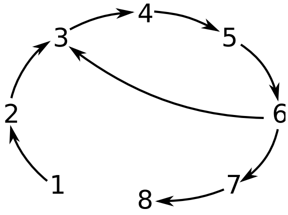
For the original rule $(1\colon S \to ABC, \{2, 5\})$, we introduce the following rules:

- $(1\colon S \to \langle 1 \rangle \$, \{X\})$
- $(X\colon \$ \to \varepsilon, \{\dots, r, s, \dots\})$
- $(r\colon \langle 1 \rangle \to ABC, \{2\})$
- $(s\colon \langle 1 \rangle \to ABC, \{5\})$

Results: Overall Nondeterminism



VS



- The first grammar is harder to analyze/parse.
- Nondeterministic rules increase complexity.
- No problem with deterministic rules.

- The first grammar is harder to analyze/parse.
- Nondeterministic rules increase complexity.
- No problem with deterministic rules.

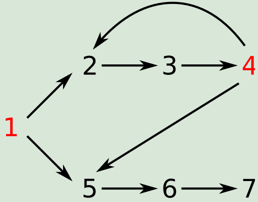
Formalization

Overall nondeterminism: the sum of the number of successors of each nondeterministic rule.

Example

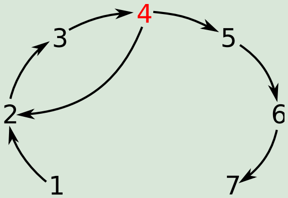
Consider $\{a^n b^n c^n \mid n \geq 1\}$.

$(1 : S \to ABC, \{2, 5\})$
$(2 : A \to aA, \{3\})$
$(3 : B \to bB, \{4\})$
$(4 : C \to cC, \{2, 5\})$
$(5 : A \to a, \{6\})$
$(6 : B \to b, \{7\})$
$(7 : C \to c, \emptyset)$



Overall nondeterminism: 4

$(1 : S \to ABC, \{2\})$
$(2 : A \to aA, \{3\})$
$(3 : B \to bB, \{4\})$
$(4 : C \to cC, \{2, 5\})$
$(5 : A \to \varepsilon, \{6\})$
$(6 : B \to \varepsilon, \{7\})$
$(7 : C \to \varepsilon, \emptyset)$



Overall nondeterminism: 2

Results: Overall Nondeterminism

Result

We cannot limit overall nondeterminism without losing generality.

Results: Overall Nondeterminism

Result

We cannot limit overall nondeterminism without losing generality.

Consider the alphabet $\Sigma = \{a_1, a_2, \ldots, a_n\}$. Define

$$L_n = \bigcup_{i=1}^{n} \{a_i\}^+$$

L_n has overall nondeterminism $n + 1$.

Open Problems

- Proof of the second result without a growing alphabet?
- Programmed grammars with appearance checking?
- Programmed grammars without ε-rules?
- Programmed grammars with leftmost derivations?

References

M. Barbaiani, C. Bibire, J. Dassow, A. Delaney, S. Fazekas, M. Ionescu, G. Liu, A. Lodhi, and B. Nagy.
The power of programmed grammars with graphs from various classes.
Journal of Applied Mathematics & Computing, 22(1–2):21–38, 2006.

H. Bordihn and M. Holzer.
Programmed grammars and their relation to the LBA problem.
Acta Informatica, 43(4):223–242, 2006.

J. Dassow and G. Păun.
Regulated Rewriting in Formal Language Theory.
Springer, New York, 1989.

A. Meduna, L. Vrábel, and P. Zemek.
On nondeterminism in programmed grammars.
In *13th International Conference on Automata and Formal Languages*, pages 316–328, Debrecen, HU, 2011. Computer and Automation Research Institute, Hungarian Academy of Sciences.

D. J. Rosenkrantz.
Programmed grammars and classes of formal languages.
Journal of the ACM, 16(1):107–131, 1969.

Discussion