LEFTIST COOPERATIVE DISTRIBUTED GRAMMAR SYSTEMS

Filip Goldefus
Doctoral Degree Programme (4), FIT BUT
E-mail: xgolde00@stud.fit.vutbr.cz

Supervised by: Alexander Meduna
E-mail: meduna@fit.vutbr.cz

Abstract: This paper discusses extension of Cooperative Distributed Grammar Systems (CDGS) with leftist-grammars as components. We investigate the descriptive power of Leftist Cooperative Distributed Grammar Systems (LCDGS) with terminating derivation mode. It arises, that LCDGS are more powerful than general CDGS with context-free grammars as components.. Especially CDGS of degree 2 with context free grammars as components working in terminating mode are as powerful as context-free grammars, contrary languages accepted by LCDGS of degree 2 are superset of context-free grammars and are included in Church Rosser Languages (CRL).

Keywords: Cooperative Distributed Grammar Systems, Leftist Grammars, Restarting Automata, Generative Power

1 INTRODUCTION

The restarting automaton was introduced by Jančar et. al. (in [2]), this model describes analysis by reduction and is capable of describing the family of Church Rosser family of languages. This paper continues with investigation of cooperative distributed grammar systems (CDGSs), which are devices consisting of several cooperating components represented by grammars or other rewriting mechanisms that work in some prescribed derivation modes (the reader is referred to [7] for more information). The components of leftist cooperative distributed grammar systems are leftist grammars (see [5]) used for decidability of accessibility problems. Using leftist grammars without insert productions, which are less powerful than context-free gammars, in cooperative distributed grammar system suprisingly led to increase of generative power. This system can be used for deciding reachability problem with multiple independent agents (components).

2 PRELIMINARIES AND DEFINITIONS

In this paper, we assume that the reader is familiar with formal language theory (see [1]) . For a finite nonempty set (an alphabet) V, let V^* represent the free monoid generated by V. Let the unit of V be denoted by ε, and let $V^+ = V^* - \{\varepsilon\}$. For $w \in V$, let $|w|$ denote the length of w, $alph(w)$ denote the set of all symbols occurring in w. The i-th symbol of w is denoted by $w[i]$, with $0 < i \leq |w|$.

A *restarting automaton* (see [3]), RRWW-automaton for short, is a construct

$$M = (Q, \Gamma, \Sigma, \delta, \rhd, \lhd, k, q_0),$$

where Q is a set of states, Γ and $\Sigma \subseteq \Gamma$ are tape and input alphabets, $\rhd, \lhd \notin \Gamma$ are the markers for the left and right border, $q_o \in Q$ is the initial state, $k \geq 1$ is a size of read/write window, and $\delta : Q \times \gamma^{\leq k} \to \mathcal{P}(Q \times (\{MVR\} \cup \gamma^{\leq k} \cup \{Restart\}) \cup \{Accept\}$, is a transition relation, $\mathcal{P}$ denotes the powerset of a set S, and $\gamma^{\leq k} = (\rhd \Gamma^{\leq k-1}) \cup \Gamma^{\leq k} \cup (\Gamma^{\leq k-1} \lhd) \cup (\rhd \Gamma^{\leq k-2} \lhd)$ denotes the set of possible contents of read/write window of M, where $\Gamma^{\leq n} = \bigcup_{i=0}^{n} \Gamma^i$.

The transition relation describes four different types of transition steps:

1. A *move-right step,* $(r,MVR) \in \delta(q,u)$, where $q,r \in Q$ and $u \in \gamma^{\leq k} - \{\lhd\}$. If M is in state q and sees the string u in its read/write window, then this move-right step causes M to shift the read/write window one position to the right and to enter the state r.

2. A *rewrite step* is of the form $(r,v) \in \delta(q,u)$, where $q,r \in Q$ and $u \in \gamma^{\leq k} - \{\lhd\}$ and $v \in \gamma^{\leq k-1}$ such that $|u| \geq |v|$. It causes M to replace the contents of u of the read/write window by the string v, and to enter state r. The read/write window is placed immediately to the right of the string v.

3. A *restart step* is of the form $(Restart) \in \delta(q,u)$, where $q,r \in Q$ and $u \in \gamma^{\leq k}$. It causes M to move its read/write window to the left end of the tape, so that the first symbol it sees is the left border marker $\rhd$, and to reenter the initial state.

4. An *accept step* is of the form $Accept \in \delta(q,u)$, where $q \in Q$ and $u \in \gamma^{\leq k}$. It causes M to halt and accept.

An input $w \in \Sigma^*$ is *accepted* by M, if there exists a computation of M which starts with the initial configuration $q_0 \rhd w \lhd$, and which finally ends with executing an accept step. By $L(M)$ we denote the language accepted by M.

A *nonforgetting restarting automaton M* (see [4]), nf-RRWW-automaton for short, consists of three phases: 1. scan the tape, 2. perform a rewrite operation, 3. execute a restart step. A *restart step* is of the form $(r,Restart) \in \delta(q,u)$, where $q,r \in Q$ and $u \in \gamma^{\leq k}$. It causes M to move its read/write window to the left end of the tape, so that the first symbol it sees is the left border marker $\lhd$, and to enter a state r. By $\mathcal{L}_{nf-RRWW}$ we will denote the class of languages that are accepted by nf-RRWW-automata. For a nf-RRWW-automaton M, the first phase of each cycle consist only of MVR-steps, that is, during this phase M behaves just like a (one-way) finite-state acceptor. Thus, the transition relation of M can be described more compactly by so-called *meta-instructions*.

A *meta-instruction* for an nf-RRWW-automaton M is either of the form $(p_1,E_1,u \to v,E_2,p_2)$ or $(p_1,E_1,Accept)$, where $p_1,p_2 \in Q$, E_1 and E_2 are regular expressions, and $u,v \in \Gamma^*$ are words satisfying $|u| \geq |v|$, which stand for a rewrite step of M. To execute a cycle M chooses a meta-instruction of the form $(p_1,E_1,u \to v,E_2,p_2)$. On trying to execute this meta-instruction M will get stuck (and so reject) starting from a configuration $p_1 \rhd w \lhd$, if w does not admit a factorization of the form $w = w_1uw_2$ such that $\rhd w_1 \in E_1$ and $w_2 \lhd \in E_2$. On the other hand, if w does have factorizations of this form, then one such factorization is chosen nondeterministically, and $p_1 \rhd w \lhd$ is transformed into $p_1 \rhd w_1vw_2 \lhd$. In order to describe the tails of accepting computations we use meta-instructions of the form $(p_1,E_1,Accept)$, where the strings from the regular language E_1 are accepted by M in tail computations, that is, without a restart. As an nf-RWW-automaton restarts immediately after executing a rewrite operation, the meta-instructions describing cycles of such an automaton are of the simplified form $(p_1,E_1,u \to v,p_2)$. By $\mathcal{L}_{nf-RWW}$ will denote the class of languages that are accepted by nonforgetting RWW-automata.

A *leftist grammar* without insert productions (with delete productions only) is a touple $G = (\Sigma,P,x)$, where Σ is a finite set of symbols, P is a finite set of productions of the form $ab \to b$, where $a,b \in \Sigma$, and a final symbol $x \in \Sigma$. A derivation step using a production $ab \to b \in P$ is defined as $uabv \Rightarrow ubv$, where $u,v \in \Sigma^*$. We say that the symbol b in the delete rule $ab \to b$ is active. As usual, the relation $\Rightarrow$ is extended to $\Rightarrow^n$, for $n \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language accepted by the leftist grammar M is defined as $L(M) = \{w \in \Sigma^* \ : \ wx \Rightarrow^* x\}$. Without loss of generality, we assume applying all productions on the leftmost possible symbols in the sentential form during a derivation (see [3]), i.e. $uabv \Rightarrow ubv$ in G using a production $ab \to b \in P$ and $\nexists cd \to d \in P$ such that $u = u_1cdu_2$, where $u,v,u_1,u_2 \in \Sigma^*$. Let $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$ be a derivation. A symbol $u_1[i]$ is alive in u_1 with respect to the derivation

$u_1 \Rightarrow^* u_p$ if there exists $j \leq i$ such that $u_1[j]$ is active with respect to $u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_p$. A symbol which is not alive is gone.

For any two strings $u, v \in V^*$, define the relation u terminally derives v in a leftist grammar G, written as $u \Rightarrow^t v$ provided that $u \Rightarrow^* v$ in G and $\nexists w \in V^*$ such that $v \Rightarrow w$ in G. A *leftist cooperating distributed grammar system* of degree n is $(n+2)$-touple $\Delta = (\Sigma, x, P_1, P_2, \ldots, P_n)$, where Σ is a finite set of symbols, $x \in \Sigma$ is a final symbol and each set $P_1, P_2, \ldots, P_n$ contains leftist productions (each component $G_i = (\Sigma, P_i, x)$ is a leftist grammar), for $i = 1, \ldots, n$. A derivation step of the i-th component using a production $ab \to b \in P_i$ is dentoed by $uabw \Rightarrow_i ubw$ for an integer $1 \leq i \leq n$ and $u, v \in \Sigma^*$, i.e. by the leftist grammar G_i.

We say that Δ accepts $w \in T^*$ in the t-mode (terminating mode), provided that there exist $m \geq 1$ and $\alpha_i \in V^*$, for $i = 1, \ldots, k$, such that $\alpha_i \Rightarrow^t \alpha_{i+1}$ in H_i, where $H_i \in \{G_1, \ldots, G_n\}$ is a component of Δ, for $i = 1, \ldots, m-1$, $\alpha_1 = wx$ and $\alpha_m = x$. Symbolically, $wx \Rightarrow^t_\Delta x$. As usual, Δ is omitted whenever the meaning is clear. The language generated by Δ in t-mode is defined as $L(\Delta, t) = \{w \in T^* \ : \ wx \Rightarrow^t_\Delta x\}$. The family of languages generated by leftist cooperating distributed grammar systems with n components working in the t-mode is denoted by $\mathcal{L}_{LD}(n)$. Let $\mathcal{L}_{CS}, \mathcal{L}_{CF}, \mathcal{L}_{DCF}$ and $\mathcal{L}_{LD}$ denote the families of context sensitive, context-free, deterministic context-free and leftist languages (accepted by leftist grammars without insert productions).

3 MAIN RESULTS

First, the following theorem proves that language family accepted by leftist grammars without insert productions are included in deterministic context-free languages family.

Theorem 1. $\mathcal{L}_{LD} \subseteq \mathcal{L}_{DCF}$.

Proof. See [3]. The proof uses simulation of a leftist grammar by a deterministic pushdown automaton. $\square$

Theorem 2. $\mathcal{L}_{LD} \not\subseteq \mathcal{L}_{REG}$.

Proof. See [3]. The proof shows that a context free language $L = \{u^n v^m \ : \ u = a_1 a_0, v = b_0 b_1, m \geq n\}$ is accepted by a leftist grammar and $L \notin \mathcal{L}_{REG}$. $\square$

Theorem 3. $\mathcal{L}_{LD}(2) \not\subseteq \mathcal{L}_{CF}$.

Proof. We describe a leftist cooperating distributed grammar system $\Delta = (\Sigma, x, P_1, P_2)$ of degree 2 working in t-mode, which generates a non-context-free language. Let $\Sigma = \{a_0, a_1, b_0, b_1, c_0, c_1\}$,

$$P_1 = \{a_1 b_0 \to b_0, b_1 c_0 \to c_0, c_1 x \to x\}$$

and

$$P_2 = \{a_0 b_0 \to b_0, b_0 c_0 \to c_0, c_0 x \to x\}.$$

Now, let $u = a_0 a_1, v = b_0 b_1$ and $w = c_0 c_1$. We make the following observation

$$u^n v^n w^n \in L(\Delta, t) \Leftrightarrow n \geq 0.$$

The implication $\Leftarrow$ is obvious. The second implication follows from two observations:

- each derivation step $x \Rightarrow^t_1 y$ of Δ deletes symbols a_1, b_1 and c_1 in the sentential form u, the rightmost symbol a_1 is deleted by the leftmost symbol b_0, the righmost symbol b_1 is deleted by the leftmost symbol c_0 and the rightmost symbol c_1 is deleted by the final symbol x;

- each derivation step $x \Rightarrow_2^t y$ of Δ deletes symbols a_0, b_0 and c_0 in the sentential form u, the rightmost symbol a_0 is deleted by the leftmost symbol b_0, the righmost symbol b_0 is deleted by the leftmost symbol c_0 and the rightmost symbol c_0 is deleted by the final symbol x.

So, the language $u^*v^*w^* \cap L(\Delta, t)$ is equal to non-context-free language $\{u^n v^n w^n \, : \, n \geq 0\}$. As the set of context-free languages are closed under intersection with a regular language, the language $L(G)$ is non-regular as well. □

Theorem 4. $\mathcal{L}_{LD}(2) \subseteq \mathcal{L}_{nf-RWW}$.

Proof. Let $\Delta = (\Sigma, x, P_1, P_2)$ be a leftist cooperating distributed grammar system. So, one can design a nonforgetting restarting automaton $M = (Q, \Gamma, \Sigma, \delta, \rhd, \lhd, k, q_0)$ such that $L(M) = L(t, \Delta)$, with $Q = \{q_0, q_1, q_2\}, \Gamma = \Sigma$, $k = 2$ and these meta-instructions:

1. if $ab \to b \in P_i$ then add (restarting) meta-instruction $(q_i, \rhd\Sigma^*, ab \to b, q_i)$ for $i \in \{1, 2\}$,
2. add meta-instructions $(q_i, \rhd\Sigma^+, x\lhd \to x\lhd, q_j)$, for $j \in \{1, 2\}$ and $i \in \{0, 1, 2\}$.
3. add meta-instructions $(q_i, \rhd x\lhd, Accept)$ for for $i \in \{1, 2\}$.

Informally, M simulates a derivation of Δ so if sentential form w of Δ contains ab, $ab \to b \in P_i$, and state of M is q_i (i-th component is active), then $w = uabv$ is rewritten by a meta-instruction constructed in (1), $(q_i, \rhd\Sigma^*, ab \to b, q_i)$, in one cycle and configuration is $q_i \rhd ubv\lhd$. If there is no applicable meta-instruction from (1), consider a configuration $q_i \rhd x\lhd$ of M, meta-instruction $(q_i, \rhd x\lhd, Accept)$ is applied, then word w is accepted. If M is in configuration $q_i \rhd ux\lhd$, where $u \in \Sigma^+$, then meta-instruction from (2) is applied and $q_i \rhd ux\lhd \Rightarrow q_j \rhd ux\lhd$, with $i, j \in \{1, 2\}$. So, the component i is switched to j, there is no meta-instruction from (1) corresponding to a production from P_i applicable.

To prove that $L(M) \subseteq L(t, \Delta)$ consider a derivation $q_i \rhd w\lhd \Rightarrow q_i \rhd u\lhd$ in M, where $q_i \in Q$ and $w, u \in \Sigma^*$. We prove that $w \Rightarrow_i^t u$ in Δ. Assume that $q_i \rhd w\lhd \Rightarrow q_i \rhd u\lhd$ by a meta-instruction $(q_i, \rhd\Sigma^*, ab \to b, q_i)$ constructed in (1), i.e. $w = w_1 ab w_2$, for $w_1, w_2 \in \Sigma^+$ and

$$q_i \rhd w_1 ab w_2 \lhd \Rightarrow q_i \rhd w_1 b w_2 \lhd$$

in Δ. Then, $w_1 ab w_2 \Rightarrow_i^t w_1 b w_2$ by i-th component of Δ using production $ab \to b \in P_i$. If $u = w$ then a meta-instruction $(q_i, \rhd\Sigma^+, x\lhd \to x\lhd, q_i)$ constructed in (2) is applied. Now, consider $q_i \rhd w\lhd \Rightarrow q_j \rhd w\lhd$ in M using a meta-instruction $(q_i, \rhd\Sigma^+, x\lhd \to x\lhd, q_j)$ constructed in (2), with $j \neq i$, then there is no meta-instruction constructed in (1) applicable and $|w| \geq 2$. This corresponds to switching from component i to j in Δ. Finally, consider configuration $q_i \rhd x\lhd$ of M, only meta-instruction $(q_i, \rhd x\lhd, Accept)$ constructed in (3) is applicable, word w such that $q_1 \rhd wx\lhd \Rightarrow q_i \rhd x\lhd$ in M is accepted, this corresponds to $wx \Rightarrow_\Delta^t x$ in Δ. Clearly, M simulates a derivation of G so that it starts by a meta-instruction constructed in (2), i.e., $q_0 \rhd wx\lhd \Rightarrow q_j \rhd wx\lhd$, for any $j \in \{1, 2\}$. The derivation then proceeds as shown above. Hence, the inclusion holds.

On the other hand, to prove that $L(t, \Delta) \subseteq L(M)$ consider a derivation $w \Rightarrow_i^t v$, where $w, v \in \Sigma^+$and $i \in \{1, 2\}$. So, $w = w_1 \Rightarrow_i \ldots \Rightarrow_i w_{n+1} = v$ using a sequence of productions $p_1, \ldots, p_n \in P_i$ in Δ. Then,

$$q_i \rhd w_1 \lhd \Rightarrow \ldots \Rightarrow q_i \rhd w_{n+1} \lhd$$

in M using meta-instructions constructed in (1) corresponding to a sequence of productions $p_1, \ldots, p_n$, i.e. $(q_i, \rhd\Sigma^*, p, q_i)$. Finally, for $v = x$, where x is final symbol, Δ accepts and $q_i \rhd x\lhd$ is accepting configuration after applying a production constructed in (3). after each derivation step $w \Rightarrow_i^t v$ component of Δ is switched, i.e. $w \Rightarrow_i^t v \Rightarrow_j^t u$, with $j \neq i$, this is realized by a meta-instruction constructed in (2). The proof now proceeds by induction. □

Corollary 5. *$\mathcal{L}_{LD}(n) \subseteq \mathcal{L}_{nf-RWW}$, for $n \geq 2$.*

Proof. Proof is similar to the proof of the previous lemma, the nonforgetting restarting automaton M is constructed in the same way, a set of states Q contains states $q_0, \ldots, q_n$ for each component of the simulated leftist cooperating distributed grammar system Δ. □

4 CONCLUSION

Denote $\mathcal{L}_{CR}$ the family of Church Rosser Languages ([6]). Recall (see [3]),

$$\mathcal{L}_L \subset \mathcal{L}_{CF} \subseteq \mathcal{L}_{LD}(2) \subseteq \mathcal{L}_{nf-RWW} \subseteq \mathcal{L}_{CR} \subseteq \mathcal{L}_{CS}.$$

5 ACKNOWLEDGEMENTS

This work was supported by the research plan "Security – Oriented Research in Information Technology", MSM 0021630528.

REFERENCES

[1] John Hopcroft and Jeffrey Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA, USA, 1979.

[2] Petr Jancar, Frantisek Mráz, Martin Plátek, and Jörg Vogel. Restarting automata. In *Proceedings of the 10th International Symposium on Fundamentals of Computation Theory*, pages 283–292, London, UK, 1995. Springer-Verlag.

[3] Tomasz Jurdziński and Krzysztof Loryś. Leftist grammars and the chomsky hierarchy. In Maciej Liskiewicz and Rüdiger Reischuk, editors, *Fundamentals of Computation Theory*, volume 3623 of *Lecture Notes in Computer Science*, pages 293–304. Springer Berlin, 2005.

[4] Hartmut Messerschmidt and Friedrich Otto. On nonforgetting restarting automata that are deterministic and/or monotone. In Dima Grigoriev, John Harrison, and Edward Hirsch, editors, *Computer Science, Theory and Applications*, volume 3967 of *Lecture Notes in Computer Science*, pages 247–258. Springer Berlin, Heidelberg, 2006.

[5] Rajeev Motwani, Rina Panigrahy, Vijay Saraswat, and Suresh Ventkatasubramanian. On the decidability of accessibility problems (extended abstract). In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, STOC '00, pages 306–315, New York, NY, USA, 2000. ACM.

[6] Gundula Niemann and Friedrich Otto. The church-rosser languages are the deterministic variants of the growing context-sensitive languages. *Inf. Comput.*, 197:1–21, February 2005.

[7] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of formal languages, vol. 2: linear modeling: background and application.* Springer-Verlag New York, Inc., New York, NY, USA, 1997.