5. Theorietag der GI Fachgruppe 0.1.5
"Automaten und formale Sprachen"

Properties of formal Languages of Therapy Plans created by Graph Grammars (short overview [1])

Daniel Kirsten
Technische Hochschule Leipzig
91 AI 2

EMail: kirsten@imn.th-leipzig.de

[1] Complete paper will be available under
http://www.informatik.th-leipzig.de/~kirsten/inf/index.html

1 Motivation

The paper deals about a special type of graph grammar used in therapy planning in complex dynamic systems in the Joint Project (BMFT-Verbundprojekt) Wiscon .
Plans are represented as finite, directed, acyclic graphs, called pin graphs.
A planner (a program) has to works on plans: creation, evaluation, simulation, revision... That means the planner has to create and to evaluate pin graphs.
Therefore Dipl.-Ing. Oksana Arnold has developed a graph grammar concept suitable for therapy planning. The core is the *rooted family* [1][2].
It is based on vertex rewriting, i.e. single vertices are replaced by complex graphs like deriving in context-free languages.
The basic idea of planning is the following: The planner starts at a pin graph called starting graph. It contains two kinds of vertices: elementary vertices and compound vertices like terminals and nonterminals in Chomsky-languages.
The planner replaces compound vertices by complex pin graphs wich may also contain compound vertices. Usually there are several pin graphs wich can replace a compound vertex. The planner inserts and evaluates pin graphs. So the search space of the planner is a set of pin graphs created by a graph grammar, or better it is a language of graphs.
The usage of graph grammars leads also to problems. Is the rooted family really suitable for therapy planning? May be there is an optimal plan to solve a problem, but the graph grammar is only able to create worse plans. May be there are several plans solving a problem, but the graph grammar is unable to create one of them. We deal therapy planning, so human life can depend on a graph grammar.
Therefore a basic research on languages creatable by rooted families was necessary! We call these languages context-free.

- How is the *power* of context-fee languages? I.e. are all recursively enumerable languages context-free, or only special cases?
- Are there interesting languages wich are not context-free ?
- Are there plans wich can not be created by a rooted family ?
- Are context-free languages stable under complement, intersection and union?
- How can I proof that a language is not contect-free?

2 Introduction

A short introduction to the basic concepts.

Definition 1.
A *hierarchically structured family of plans* is a finite set of pin graphs where:
$\mathcal{G}_i = [V_i, E_i, P_i^{in}, P_i^{out}, C_i, sub_i]$ $(i = 1 \ldots k)$, such that for all $i \in \{1 \ldots k\}$:

1. $[V_i, E_i]$ is a nonempty, finite, directed, acyclic graph.
2. $P_i^{in} := \{v | v \in V_i \quad \wedge \quad \neg\exists u \in V_i((u,v) \in E_i)\}$
   $P_i^{out} := \{v | v \in V_i \quad \wedge \quad \neg\exists u \in V_i((v,u) \in E_i)\}$
3. $C_i \subseteq V_i$
4. $sub_i : C_i \rightarrow (2^{\{1\ldots k\}} \setminus \{\emptyset\})$ $\square$

We assume that the vertices are letters of an alphabet !

Substitutions:
given: $\mathcal{F} := \{\mathcal{G}_1 \ldots \mathcal{G}_k\}$
$\mathcal{G} := \mathcal{G}_i[c \hookleftarrow \mathcal{G}_j]$
Presumption: $c \in V_i$
Definition of $\mathcal{G}$:
1. $V := (V_i \setminus \{c\}) \cup c.V_j$
2. $E := ((E_i \cup c.E_j) \setminus (V_i \times \{c\}) \setminus (\{c\} \times V_j) \cup (\{v | (v,c) \in E_i\} \times c.P_j^{in})$
   $\cup (c.P_j^{out} \times \{v | (c,v) \in E_i\})$
3. P^{in} , P^{out} already defined.
4. $C := (C_i \setminus \{c\}) \cup c.C_j$
6. $sub := sub_{j/C_i \setminus \{c\}} \cup c.sub_j$

$c.V$ simply means set a c in front of each vertex in V . This is necessary to get disjoint sets of vertices. So the vertices grow to words.
Substitution steps are commutativ and associativ. We may denote several substitutions (paths) by $\mathcal{G}[c_1 \hookleftarrow \mathcal{G}_3][c_2 \hookleftarrow \mathcal{G}_6][c_1c_3 \hookleftarrow \mathcal{G}_{42}]$

The deriving relation is defined based on substitution:

$$\mathcal{G}_i \Rightarrow_{\mathcal{F}} \mathcal{G} \quad \iff \quad \exists c \in \mathcal{G}_i \; \exists j \in sub_i(c) \; : \; \mathcal{G} = \mathcal{G}_i[c \hookleftarrow \mathcal{G}_j]$$

The transitive closure of $\Rightarrow_{\mathcal{F}}$ is $\Rightarrow_{\mathcal{F}}^{+}$.

Definition 2.
A *rooted family* is a pair $[\mathcal{F}, \mathcal{G}_i]$ such that $\mathcal{F}$ is a *hierarchically structured family of plans* and $\mathcal{G}_i$ is a pin graph in $\mathcal{F}$. $\mathcal{G}_i$ is the starting pin graph.

The *language of a rooted family* is defined by:
$L(\mathcal{R}) := \{\mathcal{G} \mid \mathcal{G}_i \Rightarrow_{\mathcal{F}}^{+} \mathcal{G} \quad \wedge \quad C = \emptyset \}$
Remember that the vertices of the created pin graphs are words: The renaming from V to $c.V$ produces longer and longer vertices. So vertices consist of several former compound vertices followed by an elementary vertex. The main point for the planning domain are the last letters. So the same plan can be represented by different graphs, there are infinite many graphs representing a given plan. This leads to a problem, solved by an equivalency concept. ($last(v)$ is simply the last letter of v.)

Definition 3.

The pin graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are equivalent iff there is a bijection $r : V_1 \to V_2$ such that :

- $\forall v \in V_1 : last(v) = last(r(v))$
- $\forall (u, v) \in (V_1 \times V_1) : (u, v) \in E_1 \longleftrightarrow (r(u), r(v)) \in E_2$

We denote it by $\mathcal{G}_1 \cong \mathcal{G}_2$.
Equivalence of pin graphs reflects the intuitive meaning of equality of plans. Equivalence is necessary, but it draws us into a tide of problems. A tasting (You find formal definitions in my paper.) :

- We have to use $\underset{\approx}{\in}$ (equivalent in) instead of $\in$. $\mathcal{G} \underset{\approx}{\in} L$ means L contains a pin graph wich is equivalent to $\mathcal{G}$.
- So we have to define new relations for subset: It is $L_1 \underset{\approx}{\subseteq} L_2$ iff all pin graphs in L_1 are equivalent in L_2 . $\underset{\approx}{\subset}$ is understood accordingly.
- Two languages are equivalent ($L_1 \cong L_2$) iff each language is an equivalent subset of the other language.
- $L_1 \underset{\approx}{\setminus} L_2$ means we remove all pin graphs from L_1 wich are equivalent in L_2.

Another problem caused by equivalence is that there are infinite languages wich contain only equivalent copies of a finite number of pin graphs. An infinite language wich can be reduced to a finite language by removing equivalent copies is called weakly infinite.

Definition 4.

A language L is called *strongly infinite* iff $\quad \nexists \hat{L} \ : \ \hat{L}$ is finite $\wedge \hat{L} \cong L_1$ $\square$

Now we are able to define the familie of context-free languages:

Definition 5. $\qquad \mathcal{PG}_{cf} := \{ \ L \mid \exists \mathcal{R} \ : \ L(\mathcal{R}) \cong L \ \}$

The set of all pin graphs over a nonempty finite alphabet Σ is defined as follows:

Definition 6.

$\mathcal{PG}^+(\Sigma) := \{\mathcal{G} \mid \quad \mathcal{G}$ is a pin graph $\quad \wedge \quad C = \emptyset \quad \wedge \quad \forall v \in V : last(v) \in \Sigma\}$ $\square$

3 A Pumping Theorem

Context-free languages allow pumping in the following way:
(You find two stronger versions in my paper.)
c^1 and c^2 are c with upper indices, so $c^2 \neq cc$!
Theorem 1.
$\forall L \in \mathcal{PG}_{cf}: \quad (L \text{ is strongly infinite}) \longrightarrow$

$\exists n \in I\!N \quad \forall \mathcal{G} \lessapprox L: \quad |\mathcal{G}| \geq n \rightarrow$

$$\begin{aligned} \exists \mathcal{G}^1, \mathcal{G}^2, \mathcal{G}^3: \quad & C^1 = \{c^1\}, |V^1| > 1, \\ & C^2 = \{c^2\}, |V^2| > 1, \\ & C^3 = \emptyset, \\ & \mathcal{G} \cong \mathcal{G}^1[c^1 \hookleftarrow \mathcal{G}^2][c^1c^2 \hookleftarrow \mathcal{G}^3], \\ & \forall j \in I\!N: \mathcal{G}^1[c^1 \hookleftarrow \mathcal{G}^2]^{j,c2}[c^1(c^2)^j \hookleftarrow \mathcal{G}^3] \lessapprox L \end{aligned}$$

□

We can not simply pump like $[c^1c^2 \hookleftarrow \mathcal{G}^2][c^1c^2 \hookleftarrow \mathcal{G}^2][c^1c^2 \hookleftarrow \mathcal{G}^2]\ldots$, because the vertex c^1c^2 will be overwritten to $c^1c^2c^2$ and so on...
We have to pump like $[c^1c^2 \hookleftarrow \mathcal{G}^2][c^1c^2c^2 \hookleftarrow \mathcal{G}^2][c^1c^2c^2c^2 \hookleftarrow \mathcal{G}^2]$.

Often it is not necessary to use the last property (the pumping) to proof that a language is not context-free .

4 Other main Results

The language of all pin graphs over an alphabet is not context-free.

Theorem 2.
$\forall\, \Sigma: \qquad \mathcal{PG}^+(\Sigma) \notin \mathcal{PG}_{cf}$ □

This is life.
A remarkable fakt is that the pumping theorem can be used to proof theorem 2. You may protest, pumping can never exceed $\mathcal{PG}^+$ because it contains each pin graph. The problem is that many pin graphs are not creatable via $\mathcal{G}^1[c^1 \hookleftarrow \mathcal{G}^2][c^1c^2 \hookleftarrow \mathcal{G}^3]$ (c.f. pumping theorem). I leave the proof because the next theorem is stronger.

L_{cd} is the language of all complete, directed graphs over the alphabet $\{a\}$. This language is not context-free. It is even worse: Each context-free language contains only a finite count of pin graphs from L_{cd} or it contains infinite many pin graphs from L_{cd} wich are only equivalent copies of finite many pin graphs from L_{cd}.

Theorem 3.
$\nexists L \in PG_{cf} \quad : \quad (L_{cd} \tilde{\cap} L)$ is strongly infinite $\square$

This theorem can be proofed using the pumping theorem.
The main lack of the rooted family is the precise rule of drawing new edges in the definition of the substitution.
Context-free languages are closed under union. The idea of the proof is easy: "Merge" the rooted families, create a new single-vertex starting graph from wich you can derive the two "old" starting graphs. This fakt and theorem 2 show that the complement of a context-free language is never context-free. However, there are languages wich are not context-free and their complements are also not context-free. To show it, you may use "the half of" L_{cd} and theorem 3. Context-free languages are not stable under intersection. An easy example is the standard example from the CHOMSKY-hierarchy, $a^n b^n c^n$.

Theorem 4.

- $\forall L_1, L_2 \in \mathcal{PG}_{cf} \quad : \quad (L_1 \cup L_2) \in \mathcal{PG}_{cf}$
- $\forall L \underset{\approx}{\subsetneq} \mathcal{PG}^+(\Sigma) \quad : \quad L$ is finite $\rightarrow L \in \mathcal{PG}_{cf}$
- $\forall L \underset{\approx}{\subsetneq} \mathcal{PG}^+(\Sigma) \quad : \quad L$ is weakly infinite $\rightarrow L \in \mathcal{PG}_{cf}$
- $\forall L \in \mathcal{PG}_{cf} \quad : \quad (\mathcal{PG}^+(\Sigma) \tilde{\setminus} L) \notin \mathcal{PG}_{cf}$
- $\exists L \subseteq \mathcal{PG}^+(\Sigma) \quad : \quad L \notin \mathcal{PG}_{cf} \quad \wedge \quad (\mathcal{PG}^+(\Sigma) \tilde{\setminus} L) \notin \mathcal{PG}_{cf}$
- $\exists \bar{L}, \hat{L} \in \mathcal{PG}_{cf} \quad : \quad (\bar{L} \tilde{\cap} \hat{L}) \notin \mathcal{PG}_{cf}$ $\square$

5 Conclusions and next Steps

The main questions are answered. But I am not able to evaluate the suitability of the rooted family for therapy planning. I figured out and proofed several properties wich allow people working in therapy planning a better understanding of the graph grammar concept they created themself. Theorem 3 looks terrible to theorists, may be practists smile about it, because they hate complete directed graphs.
Interesting problems to investigate are the following:

- The concept of the rooted family has been extended such that vertices carry logical constraints wich are inherited in substitutions. The constraints allow a shrinking of the substitution mapping...
- How important is connectity? I believe to know that it is useful to allow disconnected graphs in rooted families, even if you wish to create only connected graphs.
- (How) can we decide membership, finitness...

References

[1] Oksana Arnold and Klaus P. Jantke. Therapy plans as hierarchically structured graphs. WISCON Report 02/94, HTKW Leipzig (FH), Fachbereich IMN, April 1994

[2] Oksana Arnold and Klaus P. Jantke. Therapy Plan Generation as Program Synthesis. in LNAI Vol. 872. pages 40-55, Springer Verlag 1994

[3] John E. Hopcroft and Jeffrey D. Ullmann. Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie. Addison-Wesley, 1979