Finite Graph-Acceptors and Regular Graph-Languages

KURT-ULRICH WITT

Lehrstuhl für Informatik III, RWTH Aachen,
Büchel 29–31, D-5100 Aachen, West Germany

We introduce a new graph-acceptor model which can be considered as a canonical generalization of finite automata from strings to labelled (selector-) graphs.

1. INTRODUCTION

Many real world problems can be modelled by labelled directed graphs. Objects or substructures are represented by nodes and the relations between the objects by edges between the related nodes. Graphs themselves are usually represented by diagrams consisting of circles representing the nodes and of directed lines connecting the circles representing the edges. For big or infinite graphs and for possibly infinite sets of graphs this representation is less suitable for storing and processing graphs in computers or for classifying the problems modelled by these graphs. For that concepts for finite specification for graphs may be very useful.

Concepts for finite specifying sets of words over some finite alphabets are known in formal language theory and automata theory. So two things might be done for graphs.

Firstly, one can try to linearize graphs, i.e., a formal language is assigned to a graph by certain linearizing schemes (Witt, 1981). Then the related languages can be treated by concepts of formal language theory (Culik II and Maurer, 1977).

Secondly, formal language theory and automata theory can be generalized from strings to graphs. Graph-grammar theory has already been fruitfully developed and several interesting results and applications were found (Claus, 1978; Nagl, 1979). But on the other hand there has not been such an advance in graph-automata. Some attempts had been made (Nagl, 1979) but had been not continued and relations between graph-automata and graph-grammars had not been deeply considered apart from Rosenfeld and Milgram (1972).

So a formal graph-language theory is far from a comprehensive representation like formal language theory, where, for example, we have on the one
hand generating devices like grammars and on the other hand equivalent accepting devices—the automata. In this paper we make a first step in this direction on the regular level.

The main problem in defining graph-acceptors is finding a data structure for labelled graphs on which an acceptor can work. For that we introduce so-called monoid spaces as a generalization of input tapes, which are the input data structures for finite automata. The monoid spaces, together with some topological tools for such spaces, become the input data structure for graph-acceptors.

It is shown that the finite graph-acceptor model we then introduce can be considered as a canonical generalization of finite automata or of generalized finite automata (Thatcher and Wright, 1968) from strings or trees to graphs. Graph-acceptors and their languages have some properties which are known from finite automata and regular languages. They are closed under union, disjunction, and complementation. The equivalence to a nondeterministic version holds, too.

Following the graph-grammar concept of Nagl (1979) we introduce a new type of graph-grammar, the so-called s-regular graph-grammar, and show that they are equivalent to finite graph-acceptors.

2. A DATA STRUCTURE FOR FINITE GRAPH-ACCEPTORS

Finite automata work on input tapes on which the input strings are stored and on which the automata reads the input from left to right (Miyano, 1980). Push-down automata have input tapes, too, and a push-down store where words can be put on the top or can be pushed down from the top.

For graph-acceptors we use the embedding of graphs in discrete spaces (Witt, 1981b), where graphs are defined by topological notions of discrete spaces.

A finite monoid generating system is a pair $M = (\Sigma, E)$ consisting of a finite set of generators $\Sigma = \{a_1, \ldots, a_g\}$ and a finite set of relations $E \subseteq \Sigma^* \times \Sigma^*$. The relation $=e \subseteq \Sigma^* \times \Sigma^*$ is defined by $vxw =_E vyw$ iff $(x, y) \in E$, $v, w \in \Sigma^*$, and $\Sigma^* := \Sigma^*/=e$. $M = (\Sigma^*, \cdot)$ with $[v]_E \cdot [w]_E = [vw]_E$ is the monoid generated by $M = (\Sigma, E)$. The length of a word $w \in \Sigma^*$ is denoted by $l(w)$ and the length of $[w] \in \Sigma^*$ is defined by $l([w]) := \min \{l(v)/v \in [w]\}$.

DEFINITION. The monoid space $\mathcal{M} = (\Sigma, E)$ generated by $M = (\Sigma, E)$ consists of the points $\Sigma^*$ and a neighborhood-topology relating to every point $w \in \Sigma^*$ its neighborhood $ND(w) := \{aw/a \in \Sigma \cup \{1\}\}$ (1 is the unit of the monoid $M$).
DEFINITION. A g-structure in a monoid space \( \mathcal{M} = (\Sigma, E) \) is a triplet \( d = (\mathcal{M}, G, N(G)) \) such that (i) \( G \subseteq \Sigma^* \), (ii) \( N(G) := \{Nd(w) \mid w \in G\} \), \( Nd(w) \subseteq ND(w) \), and (iii) if \( w \in G \) and \( aw \in Nd(w) \), \( a \in \Sigma \), then \( aw \in G \). \( \Gamma(\mathcal{M}) \) denotes the set of all g-structures in \( \mathcal{M} \). \( d_e := (\mathcal{M}, \{1\}, \emptyset) \) is called the unit of \( \Gamma(\mathcal{M}) \).

G-structures are (edge-) labelled graphs embedded in monoid spaces. The node set is represented by the subset \( G \) of the space and the edges are specified by their neighborhoods, or with other words a g-structure consists of a set of neighborhoods glued together with respect to the underlying monoid space.

In the sequel we shall restrict our considerations to selector-graphs, since in computer science (data structures, program schemata (Ehrich, 1976; Six, 1981, Wegener, 1972)) interest is mostly in selector-graphs and in Witt (1981a) it is shown that labelled graphs can weakly be modified such that their modifications become selector-graphs. Selector-graphs are edge labelled graphs such that all outgoing edges from a node are labelled differently and they possess a special node, the root, from which all other nodes a reachable.

DEFINITION. Let \( d = (\mathcal{M}, G, N(G)) \) be a g-structure in \( \mathcal{M} = (\Sigma, E) \). A sequence of points \( v_1 \ldots v_m \), \( v_i \in G \), \( 1 \leq i \leq m \), is called an \( Nd \)-path in \( d \) iff \( v_{i+1} \in Nd(v_i) \), \( 1 \leq i < m \). \( NdR(v) := \{v\} \cup \{w \in G \mid \text{there is an } Nd \text{-path from } v \text{ to } w \text{ in } d\} \).

\( d \) is called an s-structure in \( \mathcal{M} \) iff \( 1 \in G \) and \( NdR(1) = G \).

\( S(\mathcal{M}) \) denotes the set of all s-structures in \( \mathcal{M} \).

DEFINITION. Let \( d = (\mathcal{M}, G, N(G)) \), \( \mathcal{M} = (\Sigma, E) \). \( w \in G \) is an almost border-point of \( d \) iff \( \forall v \in Nd(w) \) with \( l(w) \leq l(v) \): \( Nd(v) = \emptyset \). \( R(d) \) denotes the set of all almost-border-points of \( d \). \( w \in G \) is called a border-point of \( d \) iff \( Nd(w) = \emptyset \). \( L(d) \) denotes the set of all border-points of \( d \).

DEFINITION. Let \( d_i = (\mathcal{M}, G_i, N(G_i)) \in S(\mathcal{M}) \), \( i = 1, 2 \). \( d_1 \) is called a \( w \)-prefix of \( d_2(d_1 \text{ pre}(w) d_2) \) iff \( \exists w \in R(d_2): G_2 = G_1 \cup (Nd(w) \cap L(d_2)) \), and \( N(G_1) = N(G_2) \setminus \{Nd(w)\} \).

Figure 1 illustrates the prefix notion. It is \( d_i \text{ pre}(v_i) d_{i-1} \), \( 1 \leq i \leq 5 \).

The prefix notion for s-structures in monoid spaces builds the “input tape” for finite graph-acceptors.
\begin{figure}
\begin{center}
\begin{tabular}{c c}
\begin{enumerate}
\item $v_1,v_2 \in R(d_0)$
\item $v_1,v_2 \in R(d_1)$, $v_1 \in L(d_1)$
\item $v_1,v_2,v_3 \in R(d_2)$, $v_1,v_2 \in L(d_2)$
\item $v_1,v_2,v_3,v_4 \in R(d_3)$, $v_1,v_2,v_3 \in L(d_3)$
\item $v_4 \in L(d_4)$
\end{enumerate}
\end{tabular}
\end{center}
\caption{Finite Graph-Acceptors}
\end{figure}
3. THE GRAPH-ACCEPTOR MODEL

**Definition.** Let $\mathcal{M} = (\Sigma, E)$ be a monoid space with $|\Sigma| = n$. A finite (sequential) graph-acceptor over $\mathcal{M}$ is a system $GA = (\mathcal{M}, S, \delta, S_0, t)$, where $S$ is a finite set of states, $S_0 \subseteq S$ is the set of start states, $t \in S$ is the final state, and

$$\delta: \bigcup_{k=0}^{n} S^k \times \bigcup_{k=1}^{n} \Sigma^k \backslash \text{Dia}(\Sigma^k) \to S$$

$$(\text{Dia}(\Sigma^k) := \{x \in \Sigma^k/\exists i, j: i \neq j \land \text{pr}_i(x) = \text{pr}_j(x)(\text{pr}_i - \text{ith projection})\})$$

is the state transition function such that

(a) $\delta((s_1, ..., s_k), (a_1, ..., a_m)) \ (=: \delta(s_k, a_m) \text{ for short})$ is defined iff $0 \leq k \leq m, m \geq 1$,

(b) for all permutations $\pi, \varphi$ of $\{1, ..., k\}$ or $\{k + 1, ..., m\}$, respectively, it is

$$\delta((s_1, ..., s_k), (a_1, ..., a_k, a_{k+1}, ..., a_m)) = \delta((s_{\pi(1)}, ..., s_{\pi(k)}), (a_{\pi(1)}, ..., a_{\pi(k)}, a_{\varphi(k+1)}, ..., a_{\varphi(m)})).$$

**Definition.** Let $\mathcal{M} = (\Sigma, E)$ be a monoid space and $S$ a finite set. The partial function $c: \Sigma^* \to S$ is called an $S$-covering of $\mathcal{M}$. The total mapping $C: S(\mathcal{M}) \to S^{2^*}$ is called a state-table over $S(\mathcal{M})$. $\mathcal{C}_\mathcal{M}$ denotes the set of all state-tables over $\mathcal{M}$.

**Definition.** A finite graph-acceptor $GA = (\mathcal{M}, S, \delta, S_0, t)$ over $\mathcal{M} = (\Sigma, E)$ defines the transition relation $\vdash_{GA} \subseteq (S(\mathcal{M}) \times \mathcal{C}_\mathcal{M})^2$ in the following way: $(d, C) \vdash_{GA} (d', C')$ iff $\exists w \in R(d)$:

(i) $d' \text{ pre}(w)d$.

(ii) $Nd(w) = \bigcup_{j=1}^{m} \{a_{ij}w\}$.

(iii) $C(d)(a_{ij}w) = s_{ij}$, \hspace{1cm} $1 \leq j \leq k, 0 \leq k \leq m$.

(iv) $\delta((s_{i1}, ..., s_{ik}), (a_{i1}, ..., a_{ik}, a_{ik+1}, ..., a_m)) = s$.

(v) $\forall j \in \{k + 1, ..., m\} \exists x: x =_e a_{ij}w \land x$ and $a_{ij}w$ are $Nd$-Paths from 1 to $x$ in $d$.

(vi) $C'(d')(v) = \begin{cases} s, & v = w \\ C(d)(v), & v \in \text{domain}(C(d)) \\ \text{undefined}, & \text{else} \end{cases}$
Definition. Let $GA = (\mathcal{M}, S, \delta, S_0, t)$ be a finite graph-acceptor over $\mathcal{M}$. Then

$$L(GA) := \{ d \in S(\mathcal{M})/ \exists C \in \mathcal{C} : C(d)(L(d)) \subseteq S_0 \land (d, C) \xrightarrow{\varepsilon}_{GA} (d_\varepsilon, C_\varepsilon) \land C_\varepsilon(d_\varepsilon)(1) = t\}$$

is called the $s$-language accepted by $GA$ over $\mathcal{M}$. $L \subseteq S(\mathcal{M})$ is called $s$-regular over $\mathcal{M}$ iff a finite graph-acceptor $GA$ over $\mathcal{M}$ exists such that $L = L(GA)$.

Examples 3.1. (a) The class $Bi$ of binary trees is $s$-regular: $B_2 = (\mathcal{M}, B, N(B))$ with $B \subseteq \{a, b\}^*$ and

$$Nd(w) = \{aw\}, \quad aw \in B$$
$$= \{bw\}, \quad bw \in B$$
$$= \{aw, bw\}, \quad aw, bw \in B$$

is called a binary tree over $\mathcal{M} = (\{a, b\}, \emptyset)$. For the finite graph-acceptor $GA = (\mathcal{M}, \{s\}, \delta, \{s\}, s), \quad \delta(s, a) = \delta(s, b) = \delta((s, s), (a, b)) = s, \quad$ we have $Bi = L(GA)$. 
(b) The class $\mathcal{L}$ of "ladders" is $s$-regular: $\mathcal{L}_k := (\mathcal{M}, L_k, N(L_k))$ with $L_k = \{ba^i / 1 \leq i \leq k\} \cup \{a^i / 0 \leq i \leq k\}$ and

$$Nd(1) = \{a\},$$
$$Nd(a^i) = \{a^{i+1}, ba^i\}, \quad 1 \leq i \leq k,$$
$$Nd(a^k) = \{ba^k\},$$
$$Nd(ba^i) = \{ba^{i-1}\}, \quad 2 \leq i \leq k,$$

is called a ladder over $\mathcal{M} = ([a, b], \{(b, b^2a)\})$; $\mathcal{L} := \bigcup_{k \geq 2} \mathcal{L}_k$.

For the finite graph-acceptor $GA = (\mathcal{M}, \{s_0, s'_0, s_1, s_2, s_3\}, \delta, \{s'_0\}, s_3)$,

$$\delta(b) = s_0$$
$$\delta(s_0, b) = s_1$$
$$\delta((s_1, s_0), (a, b)) = s_1$$
$$\delta((s_1, s'_0), (a, b)) = s_2$$
$$\delta(s_2, a) = s_3$$

we have $\mathcal{L} = L(GA)$. 

![Figure 3](image-url)
The following example shows that there may be finite graph-acceptors with $S_0 = \emptyset$ but $L(GA) \neq \emptyset$.

**Example 3.2.** $\mathcal{M} = (\{a, b, c, r\}, \{(r, ra), (r, rb), (1, cr)\})$, $\mathcal{B}r := (\mathcal{M}, Br, N(Br))$ with $Br \subseteq \{r\} \cup \{a, b\}^*$ and $Nd(1) = \{r, a, b\}$, $Nd(r) = \{1\}$.

$$Nd(w) = \{r, xw\}, \quad w, xw \in Br, x \in \{a, b\}$$

$$= \{r\}, \quad w \in Br, xw \notin Br, x \in \{a, b\}$$

$$= \{r, aw, bw\}, \quad w, aw, bw \in Br$$

is called a *binary tree with reset* (cf. Fig. 4).

Let $\mathcal{B}re_0$ be the class of binary trees with reset in $\mathcal{M}$. For all $Br \in \mathcal{B}re_0$ it is $L(Br) = \emptyset$. Hence for a finite graph-acceptor $GA$ with $\mathcal{B}re_0 = L(GA)$ $S_0 \neq \emptyset$ may be chosen. $(GA = (\mathcal{M}, \{s, s'\}, \delta, \emptyset, s), \delta(c) = s', \delta(r) = \delta((s, s), (a, b, r)) = \delta((s, (a, r), (b, r)) = \delta((s, s, s'), (a, b, r)) = s$ accepts $\mathcal{B}re_0 \setminus \{d_e\}$.)

**Definition.** A nondeterministic (sequential) graph-acceptor over $\mathcal{M} = (\Sigma, E)$, $|\Sigma| = n$, is a system $GA_{nd} = (\mathcal{M}, S, \delta, S_0, t)$, where $S$, $S_0$, $t$ are the sets of states, start-states, and the final state and $\delta: \bigcup_{k=0}^{n} S^k \times \bigcup_{k=1}^{n} \Sigma^k \setminus \text{Dia}(\Sigma^k) \rightarrow P(S) \setminus \emptyset$ is the state transition relation ($P(S) :=$ power set of $S$) satisfying the conditions (a) and (b) for the transition function of finite graph-acceptors.

$L(GA_{nd})$ and $L(GA_{nd})$ are defined analogously to the deterministic version.
THEOREM 3.1. For every nondeterministic graph-acceptor $GA_{nd}$ over $\mathcal{M}$ a finite graph-acceptor $GA$ over $\mathcal{M}$ exists such that $L(GA_{nd}) = L(GA)$.

Proof. As in the proof of the analogous assertion for finite automata the subset-construction is used for the state set of $GA$, i.e., the reachable states of a state transition of the nondeterministic acceptor become one state in the finite acceptor. For a detailed proof we refer to Witt (1981b).

Remark 3.1. Extending the graph-acceptor model by replacing the final state by a set of final states does not mean an extension of its power (Witt, 1981b).

The following properties of $s$-regular graph-languages give reasons for calling our graph-acceptor model a canonical generalization of finite automata or of generalized finite automata (Thatcher and Wright, 1968) from strings or trees to graphs.

DEFINITION. Let $\Sigma$ be a finite alphabet, $w = a_1 \ldots a_k \in \Sigma^*$, $a_i \in \Sigma$, $1 \leq i \leq k$, $k \geq 0$; $w_i := a_1 \ldots a_i$, $1 \leq i \leq k$. The $g$-structure $d_w := (\mathcal{M}, \{1, w_1, \ldots, w_k\}, N(\{1, w_1, \ldots, w_k\}))$ over $\mathcal{M} = (\Sigma, \emptyset)$ with $\text{Nd}(1) = \{w_1\}$, $\text{Nd}(w_i) = \{w_{i+1}\}$, $1 \leq i \leq k - 1$, $\text{Nd}(w_k) = \emptyset$ is called the word-path of $w$ in $\mathcal{M}$.

Obviously we have $\forall w \in \Sigma^* : d_w \in S(\mathcal{M})$, $\mathcal{M} = (\Sigma, \emptyset)$ and $d_{\Sigma^*} := \bigcup_{w \in \Sigma^*} d_w \subseteq S(\mathcal{M})$.

THEOREM 3.2. A formal language $L$ over a finite alphabet $\Sigma$ is regular iff the set $d_L := \bigcup_{w \in L} d_w$ of word-paths of $L$ is $s$-regular over $\mathcal{M} = (\Sigma, \emptyset)$.

Proof. Let $A = (\Sigma, S, \delta, s_0, F)$ be a finite automata with $L = L(A)$. For the following nondeterministic graph-acceptor $GA_{nd} = (\mathcal{M}, S, \delta', F, s_0)$ over $\mathcal{M} = (\Sigma, \emptyset)$ with $\delta'(s, a) := \{s' \in S / \delta(s', a) = s\}$ it is $d_L = L(GA_{nd})$.

"$\subseteq$". $d_w \in d_L \Rightarrow w := a_1 \ldots a_k \in L$, $a_i \in \Sigma$, $1 \leq i \leq k$, $\Rightarrow \delta^*(s_0, w) \in F \Rightarrow \exists s_i \in S$, $0 \leq i \leq k$: $\delta(s_i, a_{i+1}) = s_{i+1}$, $0 \leq i < k$, and $s_k \in F \Rightarrow s_i \in \delta'(s_{i+1}, a_{i+1})$, $0 \leq i < k$, $\Rightarrow (\text{since } L(d_w) = \{w\}$ we may choose $C(d_w)(w) = s_k \in F$) $(d_w, C) \Rightarrow^{*}_{C_{nd}} (d_e, C_e)$ and $C_e(d_e)(1) = s_0 \Rightarrow d_w \in L(GA_{nd})$.

"$\supseteq$" and the reverse direction of the theorem follow by analogous argumentations.

THEOREM 3.3. Let $L_1$, $L_2$ be $s$-regular languages over $\mathcal{M}$. Then (a) $L_1 \cup L_2$, (b) $S(\mathcal{M}) \setminus L_1$, (c) $L_1 \cap L_2$ are $s$-regular languages over $\mathcal{M}$.

Proof. (a) Let $GA_i = (\mathcal{M}, S_i, \delta_i, S_{0i}, t_i)$ be finite graph-acceptors over $\mathcal{M}$ with $L_i = L(GA_i)$, $i = 1, 2$, where $S_1 \cap S_2 = \emptyset$. The nondeterministic
FINITE GRAPH-ACCEPTORS

Graph-acceptor $G A' = (\mathcal{M}, S, \delta, S_0, t)$ with $S := S_1 \cup S_2 \cup \{t\}, t \in S_1 \cup S_2$, $S_0 := S_{01} \cup S_{02}$, and

$$
\delta(\phi_{ik}, \alpha_m) := \begin{cases} 
\{s_i\}, & \text{if } \delta(\phi_{ik}, \alpha_m) = s_i \neq t_i \\
\{t, t_i\}, & \text{if } \delta(\phi_{ik}, \alpha_m) = t_i 
\end{cases}
$$

$\phi_{ik} \in \bigcup_{k=1}^{n} S_i^k, i = 1, 2.$

obviously accepts $L(G A_1) \cup L(G A_2)$ over $\mathcal{M}$.

(b) The extended (cf. Remark 3.1) finite graph-acceptor $G A'_1 = (\mathcal{M}, S_1, \delta_1, S_{01}, S_1 \{t_1\})$ obviously accepts $S(\mathcal{M}) \setminus L(G A_1)$.

(c) follows immediately from (a) and (b).

4. REGULAR GRAPH-LANGUAGES

Using the algorithmic graph-grammar theory due to Nagl (1979) we introduce a new type of graph-grammar the so called s-regular graph-grammar. Presuming that the reader is familiar with the basic notions of Nagl (1979) we only repeat the most necessary ones in the following.

A node- and edge-labelled graph (1-graph) over two finite alphabets $\Sigma_v, \Sigma_E$ is a triplet $d = (K, (\rho_a)_{a \in \Sigma_E}, \beta)$ consisting of a finite node set $K$, the set of labelled edge relations $\rho_a \subseteq K \times K, a \in \Sigma_E$, and the node-labelling function $\beta: K \to \Sigma_v$. $d(\Sigma_v, \Sigma_E)$ denotes the set of all l-graphs over $\Sigma_v, \Sigma_E$, $d^{1_{\text{-kno}s}}(\Sigma_v, \Sigma_E) := \{d \in d(\Sigma_v, \Sigma_E) \mid |K| = 1 \land \rho_o = \emptyset, a \in \Sigma_E\}$ is the set of all l-graphs consisting of exactly one node, and $d_{\emptyset}$ denotes the empty l-graph ($K = \emptyset$).

$\text{Gr}(d) := (K, (\rho_a)_{a \in \Sigma_E})$ is called the edge-labelled graph of $d$.

A (sequential) graph-grammar is a system $G = (\Sigma_v, \Sigma_E, A_v, A_E, d_0, P, \rightarrow_s)$ consisting of (i) finite alphabets $\Sigma_v, \Sigma_E$, (ii) terminal alphabets $A_v \subseteq \Sigma_v, A_E \subseteq \Sigma_E$, (iii) the start graph $d_0 \in d(\Sigma_v, \Sigma_E) - (d(A, \Sigma_E) \cup \{d_{\emptyset}\}$, (iv) a finite set of graph-productions $p = (d_l, d_r, E)$ (left side, right side, embedding scheme), (v) the sequential derivation scheme $-s\to$ due to Nagl (1979).

By restricting left side, right side or the embedding scheme of graph-productions graph-grammar types are defined and a graph-language hierarchy is given in Nagl (1979).

Now we introduce a new graph-grammar type which can be considered as a canonical generalization of regular string grammars.

**Definition.** An l-graph $d^* = (K, (\rho_a)_{a \in \Sigma_E}, \beta)$ over $\Sigma_v, \Sigma_E$ is called a star iff (i) $\exists k \in K: \rho_a \subseteq \{k\} \times K, \forall a \in \Sigma_E$, (ii) $\forall k' \in K \setminus \{k\} \exists a \in \Sigma'_E: (k, k') \in \rho_a$, (iii) $\forall k', k'' \in K, k' \neq k''$ and $(k, k') \in \rho_a \land (k, k'') \in \rho_b \Rightarrow a \neq b$.

$k$ is called the center of $d^*$ and $d^*(\Sigma_v, \Sigma_E)$ denotes the set of all stars over $\Sigma_v, \Sigma_E$. 
The $l$-graphs drawn in Fig. 5 are stars over $\{s_0, \ldots, s_n\}, \{a_1, \ldots, a_n, a, b, c, d, e\}$.

**Definition.** A graph-production $p = (d_t, d_r, E)$ over $\Sigma_v, \Sigma_E$ is called $s$-regular iff (i) $d_t$ consists of exactly one node ($d_t \in d^{1-kn}(\Sigma_v, \Sigma_E)$), (ii) $d_r$ is a star over $\Sigma_v, \Sigma_E$ ($d_r \in d^s(\Sigma_v, \Sigma_E)$), (iii) the embedding scheme $E = (l_a, r_a)_{a \in \Sigma_a}$ is as follows: $l_a = (L_a(k); k)$, $a \in \Sigma_E$, $r_a = (k; (1 = R_v L_{wx})(k))$, $a \in \Sigma_E \setminus \{a \in \Sigma_E/p_a^* = \emptyset\}$, $x \in \Sigma_E$, $v, w \in \Sigma_E^*$, and $k$ is the center of $d_r$.

**Definition.** A graph-grammar $G = (\Sigma_v, \Sigma_E, \Delta_v, \Delta_E, d_0, P, s\rightarrow)$ is called $s$-regular iff all productions $p \in P$ are $s$-regular and for the center $k$ of $d_r$ it is $\beta_{d_t}(k) \in \Delta_v$ and for all other nodes $k'$ of $d_r$ it is $\beta_{d_r}(k') \in \Sigma_v - \Delta_v$.

**Remarks.** (a) If in a production step a non-terminal labelled edge is generated the resulting graph does not belong to the language of the grammar, because edges cannot be manipulated by $s$-regular productions.

For that and because of we only consider sequential graph-grammars in this paper in the following $s$-regular graph-grammars are denoted as follows: $G = (\Sigma_v, \Delta_E, \Sigma, P, d_0)$, where $\Sigma$ is the (terminal) edge-labelling alphabet.

(b) $S$-regular productions do not generate ingoing edges into the right sides apart from the identical embedding of the center.

(c) $S$-regular productions at most generate outgoing edges from the center of the right sides into the host graph.

$S$-regularity can be considered as a generalization of regularity in the string case because a formal language $L$ is regular iff the set $L_{wp}$ of word-paths (words represented as $l$-graphs) of $L$ is regular (Witt, 1981b, cf. Theorem 3.2).

An $l$-graph $d$ is a selector-graph iff (i) $d$ possesses a root-node from which all other nodes can be reached, and (ii) all outgoing edges of a node of $d$ are labelled differently.
Theorem 4.1. If $G$ is an s-regular graph-grammar, then $d$ is a selector-graph for all $d \in L(G)$.

Proof. The proof follows by induction on the number of derivation steps (Witt, 1981b).

In Witt (1981a) linearizing schemes were introduced for labelled graphs. A linearizing scheme assigns a formal language over the edge-labelling alphabet to a labelled selector-graph, i.e., to every node of the graph the word-path from the root-node is assigned to.

Theorem 4.2. For every s-regular graph-grammar $G = (\Sigma, A, \Sigma, P, d_0)$ the linearization of $L(G)$ is a regular formal language over $\Sigma$.

Proof. For every s-regular productions $p \in P$ a set of regular string productions is constructed. Figure 6 illustrates roughly the idea.

The graph-production in Fig. 6 determines the following set of string productions: $A \rightarrow a_1B_1 | a_2B_2 | \ldots | a_kB_k$.

For a detailed proof we refer to Witt (1981).

Remark. The reverse of Theorem 4.2 does not hold. If we consider the class of full binary trees $Bif$, where the outgoing edges of each inner node are labelled with symbols $a$ and $b$ for example, the linearization of $Bif$ is $\{a, b\}^*$, hence regular, but we see by a pumping-lemma-like argument that $Bif$ is not s-regular (Witt, 1981b).

Let $G$ be an s-regular graph-grammar. Then $\text{Leb}(G)$ consists exactly of all graphs $d \in L(G)$, where in the derivation of $d$ a production $p$ may only be applied if the $r$-embedding component of $p$ is not empty by interpretation.

Let $d = (\mathcal{M}, G, N(G))$ be a $g$-structure in $\mathcal{M} = (\Sigma, E)$. Then $\gamma(d) := (G, (\rho_a)_{a \in \Sigma})$, where $(v, w) \in \rho_a$ if $w = av \land w \in N\!d(v), a \in \Sigma$, is the graph of $d$.

The following theorems show the equivalence of certain s-regular graph-grammars and finite graph-acceptors.

Theorem 4.3. Let $GA = (\mathcal{M}, S, \delta, S_0, t)$ be a finite graph-acceptor over $\mathcal{M} = (\Sigma, E)$ such that (i) if $aw =_E bw$, then $(a, b) \in E$, and (ii) if $w =_E aw$,
then \((1, a) \in E\). Then there exists an s-regular graph-grammar \(G\) with
\[
\bigcup_{d \in \text{Leb}(G)} \{\text{Gr}(d)\} = \bigcup_{d \in L(GA)} \{\gamma(d)\}.
\]

**Proof.** We construct an s-regular graph-grammar \(G = (\Sigma_V, \Delta_V, \Sigma, P, d_0)\) in the following way:

(a) \(\Sigma_V := S, \Delta_V := \{\bar{s}/s \in S\}\)

(b) \(d_0:\)

(c) \(P\) consists of:

1. for all \(s \in S_0\):

\[
\begin{aligned}
\bar{s} &::= \bar{s} \\
\end{aligned}
\]

2. if \(\delta((s_{i1}, \ldots, s_{ik}), (a_{i1}, \ldots, a_{ik}, a_{ik+1}, \ldots, a_{im})) = s\), then \(p = (d_1, d_r, E_p) \in P\) with \(j + 1\)

\[
\begin{aligned}
\bar{s} &::= \bar{s} \\
\end{aligned}
\]

and if \((a_{ij}, a_{il}) \in E, j \neq 1, 1 \leq j, 1 \leq k\), then the nodes \(\bar{s}_{lj}\) and \(\bar{s}_{ll}\) are identified; and if \((1, a_{il}) \in E, k + 1 \leq j \leq m\), then \(\{1,1\} \in \rho_{d_r}^{d_{j1}}\).

(\(\gamma\) ) \(l_a = (L_a(1); 1) \in E_p\) for all \(a \in \Sigma\), and if \(v =_E a_{ij}w, w \in \Sigma^*\{\varepsilon\}, v \in \Sigma^*, k + 1 \leq j \leq m\), then \(r_{au} = (1; (1 = R_vL_{a_{ij}})(1)) \in E_p\) (\(\bar{w}\) := mirror of \(w\)).

**Assertion.** \(\bigcup_{d \in \text{Leb}(G)} \{\text{Gr}(d)\} = \bigcup_{d \in L(GA)} \{\gamma(d)\}\).

\(d \in L(GA) \iff \exists h \in \mathbb{N} \exists C \in \mathbb{E}_r:\)

\[
C(d)(L(d)) \subseteq S_0 \land (d, C) \vdash_{GA}^{h} (d_e, C_e) \land C_e(d_e)(1) = t
\]

\((h = 0): C(d)(1) = t \in S_0 \Rightarrow \bar{t} :: = \bar{t} \in P \Rightarrow \bar{t} \in \text{Leb}(G))\)
\[ \exists h \in \mathbb{N} \ \exists C \in \mathcal{C}_c : C(d)(L(d)) \subseteq S_0 \land \\
(d, C) =: (d_h, C_h) \land (d_{q+1}, C_{q+1}) \vdash_{\sigma_A} (d_q, C_q), \ 0 \leq q < h, \land \\
(d_0, C_0) := (d, C) \land C_0(1) = t \]

\[ \exists h \in \mathbb{N} \ \exists C \in \mathcal{C}_c : C(d)(L(d)) \subseteq S_0 \land \\
\exists w_{q+1} R(d_{q+1}): \\
\begin{align*}
(i) & \quad d_q \text{ pre}(w_{q+1}) d_{q+1}, \\
(ii) & \quad Nd(w_{q+1}) = \bigcup_{j=1}^m \{ a_{ij} w_{q+1} \}, \\
(iii) & \quad C_{q+1}(d_{q+1})(a_{ij} w_{q+1}) = s_{ij}, 1 \leq j \leq k, 0 \leq k \leq m, \\
(iv) & \quad \delta(\sigma_k, a_{im}) = s, \\
(v) & \quad \forall j \in \{ k+1, \ldots, m \} \ \exists x: x =_E a_{ij} w_{q+1} \land x \text{ and } a_{ij} w_{q+1} \text{ are } Nd\text{-paths} \\
\end{align*} \\
\text{from 1 to } x \text{ in } d_{q+1} \\
\begin{align*}
(vi) & \quad C_q(d_q)(v) = \begin{cases} 
 s, & \quad v = w_{q+1} \\
 C_{q+1}(d_{q+1})(v), & \quad v \in \text{domain}(C_{q+1}(d_{q+1})) \\
 \text{undefined}, & \quad \text{else} 
\end{cases} \\
0 \leq q < h \land C_0(1) = t \\
\] 

\[ \implies \exists h \in \mathbb{N} : \gamma(d_q) - s \rightarrow \gamma(q_{q+1}), 0 \leq q < h, \land \gamma(d_0) = d_0 \land \gamma(d_h) - s^* \rightarrow \gamma(d') \]

by terminal productions like (c) 1. above for all \( s \in C(d_h)(L(d_h)) \), where \( \gamma(d_h) \) is isomorphic to \( \gamma(d') :=: d \in \text{Leb}(G) \).

**EXAMPLE.** The application of the above construction to Example 3.2 yields the following graph-grammar:

\[ \begin{align*}
\begin{array}{c}
1 \\
\end{array} \\
\begin{array}{c}
d_0: \\
\end{array} \\
\begin{array}{c}
\Sa \\
\end{array} \\
\begin{array}{c}
P: \\
\Sa \ ::= \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
2 \Sa \\
3 \Sa \\
4 \Sa \\
5 \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\begin{array}{c}
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\Sa \Sa \\
\end{array} \\
\end{array} \\
\begin{array}{c}
r_x = (1; (1=R_x L_a) \cup (1=R_x L_b)) (1) \\
\end{array} \\
\begin{array}{c}
r_x = (1; (1=R_x L_a) \cup (1=R_x L_b)) (1) \\
\end{array} \\
\begin{array}{c}
r_x = (1; (1=R_x L_a) \cup (1=R_x L_b)) (1) \\
\end{array} \\
\begin{array}{c}
r_x = (1; (1=R_x L_a) \cup (1=R_x L_b)) (1) \\
\end{array} \\
\begin{array}{c}
r_x = (1; (1=R_x L_a) \cup (1=R_x L_b)) (1) \\
\end{array} \\
\begin{array}{c}
r_x = (1; (1=R_x L_a) \cup (1=R_x L_b)) (1) \\
\end{array} \]
and for all productions $l_x = (L_x(1); 1), x \in \{a, b, r\}$.

**Definition.**
(a) A (selector-) graph $d$ is called embeddable into a monoid space $\mathcal{M} = (\Sigma, E)$ iff $\exists d' \in S(\mathcal{M})$: $d$ is isomorphic to $\gamma(d')$.

(b) A set of (selector-) graphs $L$ is embeddable into a monoid space $\mathcal{M} = (\Sigma, E)$ iff $\forall d \in L \exists d' \in S(\mathcal{M})$: $d$ is isomorphic to $\gamma(d')$.

**Theorem 4.4.** Let $G = (\Sigma_V, \Delta_V, \Sigma, P, d_0)$ be an $s$-regular graph-grammar such that $L(G)$ is embeddable into $\mathcal{M} = (\Sigma, E)$. Then there exists a finite graph-acceptor $G_A$ over $\mathcal{M}$ such that

$$\bigcup_{d \in L(G)} \{\text{Gr}(d)\} = \bigcup_{d \in L(G_A)} \{\gamma(d)\}.$$

**Proof.** We construct $G_A = (\mathcal{M}, S, \delta, S_0, t)$ as follows:

(a) $S := \Sigma_V$,

(b) $S_0 := \{s \in S/ (\stackrel{1}{\circ}) \cdots \stackrel{1}{\circ} (\stackrel{1}{\circ}) \in P\}$,

(c) $t := \beta_d(k_0), k_0$ is the node of $d_0$,

(d) $\delta((s_{i1}, \ldots, s_{ik}), (a_{i1}, \ldots, a_{ik}, a_{ik+1}, \ldots, a_{im})) = S$ iff $p = (d_1, d_r, E^p) \in P$ with

1. $\beta_{d_1}(k_1) \in S$

2. $d_r$: 

$$\begin{array}{c}
\text{1. } \beta_{d_1}(k_1) \in S \\
\text{2. } d_r:
\end{array}$$
and either \((1, 1) \in \rho_{a_{ij}}^d r_j \) for \(j \in \{k + 1, \ldots, m\}\) or \(r_{a_{ij}} = (1; (1 = R_v L_a)(1)) \in E^p\),
if \(v = a_{ij} w\).

The assertion \(\bigcup_{d \in L_e(G)} \{\text{Gr}(d)\} = \bigcup_{d \in L_e(G)} \{\gamma(d)\}\) is proved as the related assertion in the proof of Theorem 4.3.

5. Conclusion

The graph-linearizing concept of embedding graphs in discrete spaces (Witt, 1981b) where graphs are defined by topological notions for such spaces leads to a new type of graph-grammar, the \(s\)-regular graph-grammar, and to a new graph-acceptor model.

A graph-structure in a discrete space is built by gluing together neighborhoods with respect to the underlying space.

\(s\)-regular graph-grammars generate selector-graphs by gluing together so-called stars. Besides the linearization (due to Witt (1981)) of \(s\)-regular graph-languages are regular formal languages.

The finite graph-acceptors introduced in this paper accept graph structures in finite monoid spaces. They can be considered as generalizations of finite automata or tree automata.

The class of so defined regular graph-languages are closed under boolean operations and finite graph-acceptors and \(s\)-regular graph-grammars are in a certain sense equivalent finite specification methods for regular graph-languages.

Topics for further research would be the definition and investigation of finite parallel graph-acceptors and their relation to parallel graph-grammars; the generalization of push-down automata, turing machines to graph-structures (cf. Schönhage, 1980) and relations to corresponding graph-grammars; applications to the theory of data structures, data base theory.

Received: October 22, 1981

References


