B. EATCS 44(1991)

TECHNICAL CONTRIBUTIONS

ON CONSTRUCTING OBSTRUCTION SETS OF WORDS

Bruno COURCELLE
Université Bordeaux I
Laboratoire d'Informatique(+)
351, Cours de la Libération
33405 TALENCE Cedex, FRANCE

INTRODUCTION

The *graph minor theorem* (Robertson and Seymour [11]) states that every minor-closed set of finite graphs is characterized by a finite, canonical set of forbidden configurations called its *obstruction set*. (This definition is relative to an ordering on graphs called *minor inclusion* that we shall denote by $\trianglelefteq$; see the appendix for a quick review of definitions). The proof of the theorem does not indicate *how* the obstruction set of a given minor-closed set of graphs can be computed. The obstruction sets of the sets of partial k-trees are explicitely known for k at most 3 (Arnborg *et al* [1]). For general k, they consist of partial (k+1)-trees. The results of Lagergren [10] provide an algorithm for obtaining them. However, this algorithm seems hard to implement.

In the present note, we briefly survey several equivalent ways of specifying minor-closed subclasses of partial k-trees, and we discuss some effectivity problems concerning these characterizations. We then consider these problems in the case of sets of words (we can consider words as graphs of a special form).

(+) Laboratoire associé au CNRS. Email : courcell@geocub.greco-prog.fr. This work has been supported by the ESPRIT-Basic Research Action 3299 ("Computing by Graph Transformation").

We denote by **OBST**(L) the obstruction set of a minor-closed set of graphs L. Hence, for instance, **OBST(PLANAR)** = {$\mathbf{K_5}$, $\mathbf{K_{3,3}}$}. (See the appendix for definitions.)

Theorem 1 [11, 4] : *Let* L *be a minor-closed set of partial k-trees.*

(1) **OBST**(L) *is finite.*

(2) L is definable by a formula φ of monadic-second order (MS) logic, and also by a hyperedge replacement (HR) graph-grammar Γ.

(3) From **OBST**(L), *one can construct φ and Γ.*

(4) From φ one can construct **OBST**(L) *and Γ.*

Assertion (1) does not use the full power of the graph minor theorem: see [11, Graph minors IV, 1990]. Assertion (4) can also be obtained by the technique of Fellows and Langston [6].

It is not known whether one can construct **OBST**(L) (or equivalently φ) from Γ. (It is known that one cannot construct **OBST**(L) when L is "only" given by a membership algorithm [5]; the proof of this fact that is given by Van Leeuwen [13, Theorem 1.21] for arbitrary sets of graphs can be adapted so as to work for sets of partial 2-trees.)

Theorem 1 seems to indicate that a MS formula contains at least as much information as a HR grammar for describing a minor-closed set of partial k-trees, and perhaps strictly more. This is actually not too surprizing. The following theorem states that a finite-state automaton contains (in general) strictly more information than a context-free grammar for describing the same regular language (and the results of Courcelle [3] show that a MS formula is somewhat like a finite-state automaton for defining sets of graphs.)

Theorem 2 (Ullian [12], Harrison [9, Section 8.4]): *There is no algorithm that, given an arbitrary context-free grammar Γ produces a finite-state automaton A such that, if* **L**(Γ) *is regular, then* **L**(A) = **L**(Γ).

We now consider the effectivity questions raised by Theorem 1, in the special case of words. A word w can be considered as a directed graph consisting of a unique path, the edges of which are labelled by the letters of the word. We shall identify the word *abbc* with the graph :

$$\bullet \xrightarrow{a} \bullet \xrightarrow{b} \bullet \xrightarrow{b} \bullet \xrightarrow{c} \bullet$$

and the empty word with the single vertex graph.

For every two words w and x, $w \trianglelefteq x$ iff w is a *subword of* x, i.e., if w is obtained from x by erasing some letters (contracting an edge corresponds to erasing a letter; the labels and directions of the noncontracted edges are of course preserved).

We shall denote by **sh**(L,L') the *shuffle* of two languages L and L', i.e., the set of words $u_1v_1 \dots u_nv_n$ such that $u_1,\dots,u_n,v_1,\dots,v_n$ are words such that $u_1 \dots u_n \in L$ and $v_1 \dots v_n \in L'$. We define from any language L the following language:

$$\mathbf{OBST}(L) := (X^*-L)-\mathbf{sh}(X^*-L,X^+) \qquad (1)$$

Let us now assume that L is *subword-closed* (i.e., contains all the subwords of all its words). Then we have:

$$L = \{w \in X^* / \text{ no subword of } w \text{ belongs to } \mathbf{OBST}(L)\}, \qquad (2)$$

and by Higman's theorem, **OBST**(L) is finite (because any two words in this language are incomparable under the subword ordering). We get from equality (2) that L is rational whenever it is subword-closed, and, since the shuffle operation preserves rationality, we obtain from equality (1) that **OBST**(L) can be computed from a finite-state automaton defining L. (This result is already known from Hains [8].)

We now assume that L is given as $\trianglelefteq(L')$ where L' is defined by a *context-free* grammar Γ'. (We denote by $\trianglelefteq(L')$ the language L' augmented with all the subwords of its words.) One can easily construct a context-free grammar Γ generating L. One can also construct **OBST**(L) from Γ' (or from Γ) by equation (1) and the following result. (The algorithm given in its proof answers a question raised in [8], and is new, to the author's knowledge.)

Theorem 3 : *From a context-free grammar defining a language* L, *one can construct a regular expression defining* $\trianglelefteq(L)$.

Proof : We first give a few definitions and state a few facts concerning sets of letters and subwords of words of L.

Let $L = \mathbf{L}(\Gamma,S)$ where Γ is a context-free grammar <X,N,P,S> (terminal alphabet, nonterminal alphabet, production rules, axiom). We assume that $\mathbf{L}(\Gamma,A) \neq \emptyset$ for all $A \in N$. For every language L we let :

$\alpha(L)$ = the set of letters (terminal symbols) occurring in L
(hence $\alpha(L) = \emptyset$ iff $L \subseteq \{\varepsilon\}$).

For $L, L' \subseteq X^*$ we have:

$$\alpha(L \cup L') = \alpha(LL') = \alpha(L) \cup \alpha(L')$$
$$\trianglelefteq(L \cup L') = \trianglelefteq(L) \cup \trianglelefteq(L')$$
$$\trianglelefteq(LL') = \trianglelefteq(L)\trianglelefteq(L').$$

For $m \in (X \cup N)^*$, we let $\mathbf{L}(\Gamma, m)$ denote the language generated by Γ from m taken as axiom, and we define:

$$\alpha(m) := \alpha(\mathbf{L}(\Gamma,m))$$

and

$$\trianglelefteq(m) := \trianglelefteq(\mathbf{L}(\Gamma,m)).$$

For $A, B \in N$, we let

$B <_1 A$ iff $A \xrightarrow[G]{+} mBm'$ for some $m,m' \in (X \cup N)^*$,

$B <_2 A$ iff $A \xrightarrow[G]{+} mBm'Bm''$ for some $m,m',m'' \in (X \cup N)^*$, and

$B \equiv_1 A$ iff $A=B$ or $A <_1 B <_1 A$.

Fact 1 : *If* $A <_2 A$ *then* $\trianglelefteq(A) = \alpha(A)^*$.

Fact 2 : *If* $A \equiv_1 B$ *then* $\trianglelefteq(A) = \trianglelefteq(B)$.

We now explain how $\trianglelefteq(A)$ can be computed for any given $A \in N$.

If $A <_2 A$ (which is decidable), then Fact 1 yields the answer.

Otherwise, we compute $\trianglelefteq(A)$ in terms of the languages $\trianglelefteq(B)$ for $B <_1 A$, $B \not\equiv_1 A$, that we may assume to be given by previously computed regular expressions.

Let $p : A \longrightarrow m$ be a production rule. We let $\mathbf{R_0}(p)$, $\mathbf{R_1}(p)$, $\mathbf{R_2}(p)$ be words defined as follows:

First case : m does not contain any nonterminal B such that $B \equiv_1 A$. We let $\mathbf{R_0}(p) := m$, and $\mathbf{R_1}(p)$, $\mathbf{R_2}(p)$ be the empty word.

Second case : m contains a unique nonterminal B with $B \equiv_1 A$ and $m = m'Bm''$. We let $\mathbf{R_1}(p) := m'$ and $\mathbf{R_2}(p) := m''$. (Since we assume that $A \not<_2 A$, the word m cannot contain two occurrences of nonterminals $\equiv_1$-equivalent to A.) In this case $\mathbf{R_0}(p)$ is the empty word.

Fact 3 : *For every A such that $A \not<_2 A$, we have :*

$$\trianglelefteq(A) = (\cup\alpha(\mathbf{R_1}(p)))^*(\cup\trianglelefteq(\mathbf{R_0}(p)))\ (\cup\alpha(\mathbf{R_2}(p)))^*$$

where the unions extend to all production rules p with lefthand side B such that $B \equiv_1 A$.

Since the words $\mathbf{R_0}(p)$, $\mathbf{R_1}(p)$, $\mathbf{R_2}(p)$ contain only nonterminals C with $C <_1 A$ and $C \not\equiv_1 A$, we have achieved our goal. □

Example : We let $N = \{A,B,C,D,E,S\}$, $X = \{a,b,c,d,e,f,g\}$ and Γ be the following grammar, written as a system of equations :

$$\begin{aligned}
S &= aAb \cup bSca \cup B \\
A &= ESE \cup D \\
B &= cDd \cup de \cup EBa \cup cCa \\
C &= aBe \cup E \\
D &= aBde \\
E &= fEgEh \cup f
\end{aligned}$$

We have :

$$A \equiv_1 S >_1 B \equiv_1 C \equiv_1 D >_1 E >_2 E.$$

We get successively :

$\trianglelefteq(E) = (f \cup g \cup h)^*$

$\trianglelefteq(B) = \trianglelefteq(C) = \trianglelefteq(D) = (a \cup c \cup f \cup g \cup h)^*(\trianglelefteq(de) \cup \trianglelefteq(E))(a \cup d \cup e)^*$

(and clearly, $\trianglelefteq(de) = \varepsilon \cup d \cup e \cup de$)

$\trianglelefteq(S) = \trianglelefteq(A) = (a \cup b \cup f \cup g \cup h)^*\ \trianglelefteq(B)\ (a \cup b \cup c \cup f \cup g \cup h)^*$

If we know that a language L given by a context-free Γ is subword-closed, then we obtain **OBST**(L) from Γ by the above theorem. Is this property decidable? Certainly not because of the following: one can construct a countable family of context-free grammars Γ that generate languages of the form either X^* or $X^*-\{w\}$ (where w is a word depending on Γ) but such that one cannot decide whether $\mathbf{L}(\Gamma) = X^*$. (See [12].) Yet, $\mathbf{L}(\Gamma)$ is subword-closed iff $L(\Gamma) = X^*$.

Acknowledgements: I thank G. Sénizergues and A. Proskurowski for helpful comments on preliminary versions of this note.

References

[1] ARNBORG S., PROSKUROWSKI A., CORNEIL D., Forbidden minors characterization of partial 3-trees, Discrete Mathematics 80 (1990) 1-19.

[2] COURCELLE B., Graph rewriting: An algebraic and logic approach, in "Handbook of Theoretical Computer Science Volume B", J. Van Leeuwen ed., Elsevier, 1990, pp.193-242. (A preliminary version appeared in Bulletin of EATCS, volume 36, October 1988.)

[3] COURCELLE B., The monadic second-order logic of graphs I, Recognizable sets of finite graphs, Information and Computation 85 (1990) 12-75.

[4] COURCELLE B., The monadic second-order logic of graphs III, Tree-decompositions, minors and complexity issues,

RAIRO Informatique Théorique et Applications, to appear.

[5] FELLOWS M., LANGSTON M., On search, decision and the efficiency of polynomial time algorithms, ACM Symp. on Theory of Computing, 1989, pp.501-512

[6] FELLOWS M., LANGSTON M., An analogue of the Myhill-Nerode theorem and its use in computing finite-basis characterization, IEEE Symp. on Foundations of Computer Science, 1989, pp.520-525.

[7] HABEL A., KREOWSKI H.J., May we introduce to you : Hyperedge replacement, Proceedings of the 3rd International Workshop on Graph Grammars, L.N.C.S. 291, Springer, 1987, pp.15-26

[8] HAINS L., On free monoids partially ordered by embedding, J. of Comb. Theor. 6 (1969) 94-98.

[9] HARRISON M., Introduction to formal language theory, Addison-Wesley 1978.

[10] LAGERGREN J., Algorithms and forbidden minors for tree-decomposable graphs, Ph.D.,Report TRITA-NA-9104, Royal Institute of Technology, Stockolm, Sweden (see also the paper by Arnborg and Lagergren in the proceedings of ICALP'91, to appear).

[11] ROBERTSON N., SEYMOUR P., Graph minors I to XVII, a series of papers published in J. Comb. Theory Ser. B, (first paper in volume 35, 1983)

[12] ULLIAN J., Partial algorithm problems for context-free languages, Information and Control 11(1967) 80-101.

[13] VAN LEEUWEN J., Graph algorithms, in "Handbook of Theoretical Computer Science, volume A", J. Van Leeuwen ed., Elsevier, 1990, pp. 525-632.

APPENDIX

A graph H is a *minor* of a graph G (or *is included in* G *as a minor*) if it can be obtained from G by a sequence of edge contractions, of edge deletions, and of deletions of isolated vertices. We denote this by $H \trianglelefteq G$. Since we only consider finite graphs up to isomorphisms (i.e., any two isomorphic graphs are considered as equal), this relation is a partial order. A set of graphs L is *minor-closed* if it contains all minors of all its elements. If this is the case:

L = {G/ no graph H in **OBST**(L) is a minor of G}

where:

OBST(L) = {G / G is a graph not in L,
and every minor of G different from G is in L}.

The set **OBST**(L) is called the obstruction set of L. The graph minor theorem (Robertson and Seymour [11]) states that **OBST**(L) is finite for every minor-closed set of graphs.

A *partial k-tree* is any subgraph of a k-tree; *k-trees* are constructed recursively as follows: the clique with k vertices is a k-tree; in order to form a k-tree with n vertices, one adds a new vertex to a k-tree T with n-1 vertices, and edges linking this new vertex to the vertices of a clique of T having k vertices. Partial k-trees can be also characterized in terms of tree-decompositions ([11]; see Van Leeuwen [13] for a proof of the equivalence of the two characterizations). Partial k-trees are important in the theory of graph algorithms (see [13]) and also because of their relations to hyperedge replacement graph-grammars. We refer the reader to Courcelle [2,3,4] or Habel and Kreowski [7] for *hyperedge replacement graph-grammars*. Let us only mention that they can be considered as an extension to graphs of context-free (word) grammars, and that every context-free set of graphs is a set of partial k-trees for some fixed k, up to loops, multiple edges and labels.

The use of *monadic second-order logic* for describing graph properties is explained in Courcelle [2,3,4].