ON CONSTRUCTING OBSTRUCTION SETS
OF WORDS

Bruno COURCELLE
Université Bordeaux 1
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 \( \preceq \); 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 explicitly known for \( k \) at most 3 (Arnborg et al [11]). 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 effectiveness 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).

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

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

1. \( \text{OBST}(L) \) is finite.
2. \( L \) is definable by a formula \( \phi \) of monadic-second order (MS) logic, and also by a hyperedge replacement (HR) graph-grammar \( \Gamma \).
3. From \( \text{OBST}(L) \), one can construct \( \phi \) and \( \Gamma \).
4. From \( \phi \) one can construct \( \text{OBST}(L) \) and \( \Gamma \).

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 \( \text{OBST}(L) \) (or equivalently \( \phi \)) from \( \Gamma \). (It is known that one cannot construct \( \text{OBST}(L) \) 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 an HR grammar for describing a minor-closed set of partial \( k \)-trees, and perhaps strictly more. This is actually not too surprising. 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 \( \Gamma \), produces a finite-state automaton \( A \) such that, if \( L(\Gamma) \) is regular, then \( L(A) = L(\Gamma) \).

We now consider the effectiveness 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:
and the empty word with the single vertex graph.

For every two words $w$ and $x$, $w \preceq x$ 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 $\text{sh}(L, L')$ the shuffle of two languages $L$ and $L'$, i.e., the set of words $u_1v_1 \ldots u_nv_n$ such that $u_1, \ldots, u_n, v_1, \ldots, v_n$ are words such that $u_1, \ldots, u_n \in L$ and $v_1, \ldots, v_n \in L'$. We define from any language $L$ the following language:

$$\text{OBST}(L) := (X^* - L) - \text{sh}(X^* - L, X')$$

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

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

and by Higman's theorem, $\text{OBST}(L)$ 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 $\text{OBST}(L)$ 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 $\preceq(L)$ where $L'$ is defined by a context-free grammar $\Gamma$. (We denote by $\preceq(L)$ the language $L'$ augmented with all the subwords of its words.) One can easily construct a context-free grammar $\Gamma'$ generating $L$. One can also construct $\text{OBST}(L)$ from $\Gamma'$ (or from $\Gamma$) 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 $\preceq(L)$.

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

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

$$\alpha(L) = \text{the set of letters (terminal symbols) occurring in } L$$

(hence $\alpha(L) = \emptyset$ iff $L \subseteq \{e\}$).

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

$$\alpha(L \cup L') = \alpha(L) \cup \alpha(L')$$

$$\preceq(L \cup L') = \preceq(L) \cup \preceq(L')$$

$$\preceq(L') = \preceq(L) \preceq(L').$$

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

$$\alpha(m) := \alpha(L(\Gamma, m))$$

and

$$\preceq(m) := \preceq(L(\Gamma, m)).$$

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

$$B <_A A \iff A \xrightarrow{\beta} mBm'$$

for some $m, m' \in (X \cup N)^*$.

$$B <_2 A \iff A \xrightarrow{\beta} mBm'Bm'$$

for some $m, m', m'' \in (X \cup N)^*$, and

$$B \equiv A \iff A = B \text{ or } A <_1 B <_1 A.$$

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

**Fact 2:** If $A \equiv B$ then $\preceq(A) = \preceq(B)$.
We now explain how $\leq(A)$ can be computed for any given $A \in N$.

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

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

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

**First case**: $m$ does not contain any nonterminal $B$ such that $B =_1 A$. We let $R_0(p) := m$, and $R_1(p)$, $R_2(p)$ be the empty word.

**Second case**: $m$ contains a unique nonterminal $B$ with $B =_1 A$ and $m = m'Bm"$. We let $R_1(p) := m'$ and $R_2(p) := m"$. (Since we assume that $A \not\leq A$, the word $m$ cannot contain two occurrences of nonterminals $=_1$-equivalent to $A$.) In this case $R_0(p)$ is the empty word.

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

$$\leq(A) = (\cup a(R_1(p)))^* (\cup \leq(R_0(p))) \cup a(R_2(p)))^*$$

where the unions extend to all production rules $p$ with left-hand side $B$ such that $B =_1 A$.

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

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

$$S = aAb \cup bSc \cup B$$
$$A = ESE \cup D$$
$$B = cDd \cup de \cup EBa \cup cCa$$
$$C = aBe \cup E$$
$$D = abde$$
$$E = \epsilon \Rightarrow E \epsilon \cup f$$

We have:

$$A \equiv_1 S \equiv_1 B \equiv_1 C \equiv_1 D \equiv_1 E$$

We get successively:

$$\leq(E) = (\epsilon \cup b \cup g \cup h)^*$$
$$\leq(B) = \leq(C) = \leq(D) = (a \cup d \cup f \cup g \cup h)^* (\epsilon \cup d e \cup \leq(E)) [a \cup d \cup d e \cup (a \cup d \cup d e)]$$

and clearly, $\leq(\epsilon) = \epsilon$.

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

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


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


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 $H \preceq G$. 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:

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

where:

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

The set $\text{OBST}(L)$ is called the obstruction set of $L$. The graph minor theorem [Robertson and Seymour [11]] states that $\text{OBST}(L)$ 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].