A Lower Bound for the Intersection of Regular Forests

by Dennis M. Volpano

October 1993

Approved for public release; distribution is unlimited.

Prepared for:

Naval Postgraduate School
Monterey, California 93943
Best Available Copy
A Lower Bound for the Intersection of Regular Forests

Dennis M. Volpano

Regular $\Sigma X$-forests continue to play an important role in programming languages, specifically in the design of type systems. They arise naturally as terms of constructor-based, recursive data types in logic and functional languages. Deciding whether the intersection of a sequence of regular $\Sigma X$-forests is nonempty is an important problem in type inference. We show that this problem is PSPACE-hard and as a corollary that the problem of constructing a regular $\Sigma X$-grammar representing their intersection is PSPACE-hard.
A Lower Bound for the Intersection of Regular Forests

Dennis M. Volpano

Department of Computer Science
Naval Postgraduate School
Monterey, California, USA

email: volpano@cs.nps.navy.mil

October 5, 1993

Abstract

Regular EX-forests continue to play an important role in programming languages, specifically in the design of type systems [MiR85, AM91, Vol93]. They arise naturally as terms of constructor-based, recursive data types in logic and functional languages. Deciding whether the intersection of a sequence of regular EX-forests is nonempty is an important problem in type inference. We show that this problem is PSPACE-hard and as a corollary that the problem of constructing a regular EX-grammar representing their intersection is PSPACE-hard.

1 Introduction

Regular EX-forests are playing an increasingly important role in language design and in particular in the design of type systems. Type inference then usually relies upon various operations over regular forests, one of which is $RF$-$INT$, deciding the emptiness of their intersection.

Definition 1.1 The problem $RF$-$INT$ is given a sequence of regular EX-grammars $G_1, \ldots, G_m$, decide whether $\bigcap_{k=1}^m T(G_k)$ is nonempty.
Regular forests have been used to characterize the types of logic and functional programs [Mis84, MiR85, HeJ90, AM91] as well as overloading introduced through classes in Haskell [Kae88, Vol93]. For example, Heintze and Jaffar propose what amounts to regular ΣX-grammars as inferred "types" or approximations of the semantics of logic programs. Corresponding to a logic program, say

\[
\begin{align*}
p(a). \\
p(f(X)) & \leftarrow p(X). \\
r(b). \\
r(f(Y)) & \leftarrow r(Y). \\
q(Z) & \leftarrow p(Z), r(Z).
\end{align*}
\]

is a set of equations

\[
\begin{align*}
X &= a \cup f(X) \\
Y &= b \cup f(Y) \\
Z &= X \cap Y
\end{align*}
\]

whose simultaneous least fixed point is an approximate meaning of the program. The inferred approximation or "type" is given by

\[
\begin{align*}
X &= a \cup f(X) \\
Y &= b \cup f(Y) \\
Z &= \emptyset
\end{align*}
\]

Solving for variable \(Z\) requires deciding whether the intersection of the two regular forests described by the first two equations is nonempty. One can also view the logic program above as describing a set of valid overloading in Haskell for \(p\) and \(r\) as operators where \(p\) has instances at types \(a\) and \(f\), and \(r\) at \(b\) and \(f\):

\[
\begin{align*}
\text{class } P \alpha \text{ where } p :: \alpha \\
\text{instance } P a \text{ where } p = \ldots \\
\text{instance } P f(X) \Rightarrow P f(X) \text{ where } p = \ldots \\
\text{class } R \alpha \text{ where } r :: \alpha \\
\text{instance } R b \text{ where } r = \ldots \\
\text{instance } R f(Y) \Rightarrow R f(Y) \text{ where } r = \ldots
\end{align*}
\]

Instance declarations for an overloaded operator in Haskell describe a regular forest. So for example, deciding whether term \(p = r\) is typable requires
deciding whether the regular forest arising from p's instance declarations intersects with the forest described by instances for r.

2 Forests and Regular $\Sigma X$-grammars

Given an alphabet $A$, an $A$-valued tree $t$ is specified by its set of nodes (the "domain" $\text{dom}(t)$) and a valuation of the nodes in $A$. Formally, a $k$-ary, $A$-valued tree is a map $t : \text{dom}(t) \to A$ where $\text{dom}(t) \subseteq \{0, \ldots, k-1\}^*$ is a nonempty set, closed under prefixes. The frontier of $t$ is the set

$$\{w \in \text{dom}(t) \mid \exists i. wi \in \text{dom}(t)\}.$$ 

It is assumed that $A$ is partitioned into a ranked alphabet $\Sigma$ and a frontier alphabet $X$. A ranked alphabet, or signature, is a finite nonempty operator domain. For any $\Sigma$ and $X$, we denote the set of all finite $\Sigma X$-trees by $F_{\Sigma}(X)$. A forest, or tree language, $T \subseteq F_{\Sigma}(X)$ is called regular if and only if for some finite set $C$ disjoint from $\Sigma$ and $X$, $T$ can be obtained from finite subsets of $F_{\Sigma}(X \cup C)$ by applications of union, concatenation $\cdot$, and closure $\ast$ where $c \in C$ [Tho90].

A regular forest can alternatively be defined as a tree language generated by a regular $\Sigma X$-grammar [Ge84].

**Definition 2.1** A regular $\Sigma X$-grammar $G$ consists of

- a finite nonempty set $N$ of nonterminal symbols,
- a finite set $P$ of productions of the form $A \rightarrow r$ where $A \in N$ and $r \in F_{\Sigma}(N \cup X)$, and
- an initial symbol $S \in N$.

**Definition 2.2** If $G = (N, \Sigma, X, P, S)$ is a regular $\Sigma X$-grammar then the $\Sigma X$-forest generated by $G$ is

$$T(G) = \{t \in F_{\Sigma}(X) \mid S \Rightarrow_{G} t\}$$

Regular $\Sigma X$-grammars are a class of context-free grammars that define the same family of forests as those recognized by nondeterministic root-to-frontier (NDR) $\Sigma X$-automata. A root-to-frontier automaton can be viewed
as an attribute evaluator for a tree whose attributes are states prescribed by an attribute grammar with inherited attributes only. Formally, a NDR $\Sigma X$-automaton $A$ is a tuple $(A, A', \alpha)$ such that

1. $A$ is a finite NDR $\Sigma$-algebra $(A, \Sigma)$,
2. $A' \subseteq A$ is a set of initial states, and
3. $\alpha : X \rightarrow pA$ is a final assignment.

In a NDR $\Sigma$-algebra $(A, \Sigma)$, $A$ is a nonempty set of states and every $\sigma \in \Sigma_\omega$ with $m \geq 1$ is realized as a mapping $\sigma^A : A \rightarrow p(A^m)$. For $\sigma \in \Sigma_0$, $\sigma^A$ is a subset of $A$.

For example, a NDR $\Sigma X$-automaton $A = (A, A', \alpha)$ recognizing set

$$\{ (x, y), (y, x) \}$$

can be defined as follows. Let $\Sigma = \Sigma_2 = \{ \sigma \}$, $X = \{ x, y \}$, and the set of initial states $A' = \{ S \}$. Define $A = (\{ \hat{x}, \hat{y}, S \}, \Sigma)$ such that

$$\sigma^A(S) = \{ (\hat{x}, \hat{y}), (\hat{y}, \hat{x}) \}$$

and finally define the final assignment $\alpha$ as

$$x\alpha = \{ \hat{y} \}$$
$$y\alpha = \{ \hat{x} \}$$

It is interesting to note that there is no deterministic root-to-frontier $\Sigma X$-automaton that accepts the set above. Suppose automaton $A$ accepts $\sigma(x, y)$ and $\sigma(y, x)$ and that $\sigma(a) = (a_1, a_2)$ for some states $a$, $a_1$, and $a_2$ of $A$. If $\alpha$ is $A$'s final assignment function, then

$$x\alpha = a_1, \ y\alpha = a_2, \ y\alpha = a_1, \ x\alpha = a_2$$

Since $A$ is deterministic, $a_1 = a_2$. So we have $\sigma(a) = (a_1, a_1)$ where $x\alpha = y\alpha = a_1$. Therefore on $\sigma(x, x)$ and $\sigma(y, y)$, $A$ enters the leaves in state $a_1$ such that $a_1 \in x\alpha$, and $a_1 \in y\alpha$. Thus $A$ accepts $\sigma(x, x)$ and $\sigma(y, y)$ as well.

Given that regular $\Sigma X$-grammars define exactly the forests recognized by NDR $\Sigma X$-automata, one could formulate $RF$-$INT$ in terms of the latter representation of regular forests. But we choose regular $\Sigma X$-grammars instead since they are better suited for manipulation.

Regular forests are effectively closed under intersection.
Theorem 2.1 If $G_1$ and $G_2$ are regular $\Sigma X$-grammars, for a given $\Sigma$ and $X$, then $T(G_1) \cap T(G_2)$ is a forest generated by a regular $\Sigma X$-grammar.

Proof. Suppose $G_1 = (N_1, \Sigma, X, P_1, S_1)$ and $G_2 = (N_2, \Sigma, X, P_2, S_2)$ are regular $\Sigma X$-grammars. Let $\Sigma X$-grammar $G = (N_1 \times N_2, \Sigma, X, P, [S_1, S_2])$ where

$$[A, B] \rightarrow a([Y_1, Z_1], \ldots, [Y_n, Z_n]) \in P, \text{ for } n \geq 0$$

if and only if

$$A \rightarrow a(Y_1, \ldots, Y_n) \in P_1,$$

$$B \rightarrow a(Z_1, \ldots, Z_n) \in P_2,$$

and $a \in \Sigma$, or $[A, B] \rightarrow a \in P$ if and only if $a \in X$. Then $T(G) = T(G_1) \cap T(G_2)$. □

The theorem implies that the family of regular forests is properly contained within the context-free languages since the latter is not closed under intersection.

We now state and prove the main result.

Theorem 2.2 $RF$-INT is PSPACE-hard.

Proof. The proof uses a result of [Koz77]. For every deterministic Turing machine $M$ of polynomial space complexity, we give a log-space transducer that on input $x$, outputs a sequence of regular $\Sigma X$-grammars whose intersection is nonempty iff $M$ accepts $x$.

Let $M$ be a single tape DTM of polynomial space complexity $p(n) \geq n$ and assume that $M$ always makes at least three odd number of moves, has a unique accepting state, $q_{acc}$, and erases its tape before accepting, positioning its tape head at the left end of the tape. Let $x = a_1 \ldots a_n$ be a string over $M$'s input alphabet and suppose $M$ has states $Q$ and tape symbols $\Gamma$ such that $Q$, $\Gamma$, and set $\{nil, \#, \#\}$ are pairwise disjoint. If

$$\Delta = \Gamma \cup \{[qX] \mid q \in Q \& X \in \Gamma\}$$

then ranked alphabet $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$ where $\Sigma_0 = \{nil\}$, $\Sigma_1 = \Delta$, $\Sigma_2 = \{\#\}$ and $\Sigma_3 = \{\#\}$. Suppose $IDA$ derives regular forest

$$Z_1(Z_2(\ldots Z_{p(n)}(nil)\ldots))$$
for all \( Z_k \in \Delta, 1 \leq k \leq p(n) \), and \( ID_{i[X_1X_2X_3]} \) derives regular forest

\[
Z_1 (\cdots Z_{i-1} (X_1 (X_2 (X_3 (Z_i (\cdots Z_{p(n)-3} (\text{nil}) \cdots)
\]

for all \( X_1, X_2, X_3, Z_k \in \Delta, 1 \leq k \leq p(n) - 3 \).

A computation of \( M \) consists of a sequence of instantaneous descriptions

\[
ID_0 \vdash ID_1 \vdash \cdots \vdash ID_{2m+1},
\]

each containing the contents of \( M \)'s tape padded with blanks (\( B \)'s) to length \( p(n) \). If according to a move of \( M \), symbols \( Y_1 Y_2 Y_3 \) in positions \( i, i+1, \) and \( i+2 \) respectively of one \( ID \) can follow from symbols \( X_1 X_2 X_3 \) in the same positions of another \( ID \), we write

\[
ID_{i[X_1X_2X_3]} \vdash_M ID_{i[Y_1Y_2Y_3]} \]

We give two regular \( \Sigma X \)-grammars \( F_{i^{\text{odd}}} \) and \( F_{i^{\text{even}}} \) such that \( F_{i^{\text{odd}}} \) ensures that even \( ID \)'s follow from odd ones, and \( F_{i^{\text{even}}} \) that odd ones follow from even ones. Let \( F_{i^{\text{odd}}} \) be a regular \( \Sigma X \)-grammar with empty frontier alphabet, start symbol \( S \) and productions

\[
S \rightarrow \#(ID_\Delta, ID_{i[Z_1Z_2Z_3]}, F_{i[Z_1Z_2Z_3]})
\]

for all \( Z_k \in \Delta, 1 \leq k \leq 3, \)

\[
F_{i[X_1X_2X_3]} \rightarrow \#(ID_{i[Y_1Y_2Y_3]}, ID_{i[Z_1Z_2Z_3]}, F_{i[Z_1Z_2Z_3]})
\]

for all \( X_k, Y_k, Z_k \in \Delta, 1 \leq k \leq 3, \) such that \( ID_{i[X_1X_2X_3]} \vdash_M ID_{i[Y_1Y_2Y_3]} \), and

\[
F_{i[X_1X_2X_3]} \rightarrow \#\#(ID_{i[Y_1Y_2Y_3]}, ID_\Delta)
\]

for all \( X_k, Y_k \in \Delta, 1 \leq k \leq 3, \) such that \( ID_{i[X_1X_2X_3]} \vdash_M ID_{i[Y_1Y_2Y_3]} \).

Let \( F_{i^{\text{even}}} \) be a regular \( \Sigma X \)-grammar with empty frontier alphabet, start symbol \( S \) and productions

\[
S \rightarrow \#(ID_{i[X_1X_2X_3]}, ID_{i[Y_1Y_2Y_3]}, S)
\]

\[
S \rightarrow \#\#(ID_{i[X_1X_2X_3]}, ID_{i[Y_1Y_2Y_3]})
\]

for all \( X_k, Y_k \in \Delta, 1 \leq k \leq 3, \) such that \( ID_{i[X_1X_2X_3]} \vdash_M ID_{i[Y_1Y_2Y_3]} \).

Finally, suppose \( \text{initID} \) derives the unary tree

\[
[q_{0}a_1](a_2(\cdots a_n(B_{n+1}(\cdots B_{p(n)}(\text{nil}) \cdots)
\]
where $B_k$ is a blank and $q_0$ is the start state of $M$, and $\text{finalID}$ derives
\[ [q_{\text{acc}}B](B_2\cdots B_{p(n)}(\text{nil})\cdots) \]

Then let $F_{\text{end}}$ be a regular grammar with start symbol $S$ and productions
\[
S \rightarrow \#(\text{initID}, ID_\Delta, F_{\text{acc}})
\]
\[
F_{\text{acc}} \rightarrow \#(ID_\Delta, ID_\Delta, F_{\text{acc}})
\]
\[
F_{\text{acc}} \rightarrow \#(ID_\Delta, \text{finalID})
\]

Then we have
\[
\bigcap_{i=1}^{p(n)-2} T(F_i^{\text{odd}})
\]
iff $u = \#(ID_0, ID_1, \#(\cdots \#(ID_{2m-2}, ID_{2m-1}, \#(ID_{2m}, ID_{2m+1})\cdots) and from $ID_{2k-1}$ follows $ID_{2k}$ according to the transition rules of $M$ for $1 \leq k \leq m$. Likewise,
\[
\bigcap_{i=1}^{p(n)-2} T(F_i^{\text{even}})
\]
iff $u = \#(ID_0, ID_1, \#(\cdots \#(ID_{2m-2}, ID_{2m-1}, \#(ID_{2m}, ID_{2m+1})\cdots) and from $ID_{2k}$ follows $ID_{2k+1}$ according to the rules of $M$ for $0 \leq k \leq m$. Then
\[
T(F_{\text{end}}) \cap \bigcap_{i=1}^{p(n)-2} T(F_i^{\text{odd}}) \cap T(F_i^{\text{even}})
\]
is nonempty iff $M$ accepts $x$. \qed

As is the case for emptiness of intersection of a sequence of DFA's, the source for the hardness of $RF$-INT lies not in deciding emptiness but rather in computing the intersection of regular forests.

**Corollary 2.3** Given regular $\Sigma X$-grammars $G_1, \ldots, G_m$, constructing a regular $\Sigma X$-grammar $G$ such that $T(G) = \bigcap_{k=1}^m T(G_k)$ is PSPACE-hard.

**Proof.** The emptiness of $T(G)$ for a regular $\Sigma X$-grammar $G$ is decidable in time $O(|G|^2)$ in the usual way. From the proof of Theorem 2.2 then every problem in PSPACE is P-time Turing reducible to the problem of constructing the intersection of a sequence of regular $\Sigma X$-grammars. \qed
A simple algorithm for constructing $G$ is based on the usual construction of forming the cartesian product of reachable states as is suggested in the proof of Theorem 2.1 [AiM91]. It has worst-case time complexity exponential in $m$. Unfortunately this naive construction is likely the best we can do. It should be pointed out that for a fixed $m$, constructing $G$ from $G_1, \ldots, G_m$ can be done in polynomial time.

Deciding whether some number of DFA's accept a common string can be done in nondeterministic linear space, but this does not appear to be true for $RF$-$INT$, which can be decided in deterministic exponential time. This suggests that a tighter lower bound exists for $RF$-$INT$.

References


Distribution List

Defense Technical Information Center  
Cameron Station  
Alexandria, VA 22314  
2

Library, Code 52  
Naval Postgraduate School  
Monterey, CA 93943  
2

Director of Research Administration  
Code 08  
Naval Postgraduate School  
Monterey, CA 93943  
1

Dr. Neil C. Rowe, Code CSRp  
Naval Postgraduate School  
Computer Science Department  
Monterey, CA 93943-5118  
1

Prof. Robert B. McGhee, Code CSMz  
Naval Postgraduate School  
Computer Science Department  
Monterey, CA 93943-5118  
1

Dr. Ralph Wachter  
Software Program  
Office of Naval Research  
800 N. Quincy St.  
Arlington VA 22217-5000  
2

Dr. Dennis Volpano, Code CSVo  
Naval Postgraduate School  
Computer Science Dept.  
Monterey, CA 93943-5118  
20