Science of Computer Programming 96 (2014) 511–526

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Monolithic and modular termination analyses for higher-order attribute grammars

Lijesh Krishnan, Eric Van Wyk [*]

Department of Computer Science and Engineering, University of Minnesota, Twin Cities, Minneapolis, MN, USA

HIGHLIGHTS

- We give a conservative non-termination analysis for higher-order attribute grammars.
- The analysis is applicable to many attribute grammar idioms.
- Our attribute grammar specification of Java 1.4 passes the analysis.
- We provide a modular version of the analysis applicable to extensible languages.

ARTICLE INFO

Article history:
Received 15 March 2013
Received in revised form 18 May 2014
Accepted 21 May 2014
Available online 11 June 2014

Keywords:
Higher-order attribute grammars
Attribute grammar analysis
Termination analysis

ABSTRACT

In this paper we describe a sound, but not complete, analysis to prove the termination of higher-order attribute grammar evaluation caused by the creation of an unbounded number of (finite) trees as local tree-valued attributes, which are then themselves decorated with attributes. The analysis extracts a set of term-rewriting rules from the grammar that model creation of new syntax trees during the evaluation of higher-order attributes. If this term rewriting system terminates, then only a finite number of trees will be created during attribute grammar evaluation. The analysis places an ordering on nonterminals to handle the cases in which higher-order inherited attributes are used to ensure that a finite number of trees are created using such attributes. When paired with the traditional completeness and circularity analyses for attribute grammars and the assumption that each attribute equation defines a terminating computation, this analysis can be used to show that attribute grammar evaluation will terminate normally. This analysis can be applied to a wide range of common attribute grammar idioms and has been used to show that evaluation of our specification of Java 1.4 terminates. We also describe a modular version of the analysis that is performed on independently developed language extension grammars and the host language being extended. If the extensions individually pass the modular analysis then their composition is also guaranteed to terminate.

1. Introduction

Attribute grammars (AGs) were introduced by Knuth [1] in 1968 as a means to assign semantics to syntax trees, by associating tree nodes with named values called attributes. Once the program syntax tree has been constructed (typically by a parser), the attribute evaluator computes values for its undefined attribute instances based on equations associated with the grammar productions. Attribute grammars are used in a number of applications such as language prototyping and

[*] Corresponding author.
E-mail addresses: krishnan@cs.umn.edu (L. Krishnan), evw@cs.umn.edu (E. Van Wyk).

http://dx.doi.org/10.1016/j.scico.2014.05.016
0167-6423/© 2014 Elsevier B.V. All rights reserved.

```
while: w::Stmt ::= cond::Expr   body::Stmt
    {   local attribute translate :: Stmt;
        translate = if ... then ...
                    else ifThen(cond, seq(body, while(cond, body)));  }

doWhile: dw::Stmt ::= body::Stmt   cond::Expr
    {   local attribute translate :: Stmt;
        translate = seq(body,  while(cond, body));  }

ifThen: it :: Stmt::: := cond::Expr   body::Stmt
    {   local attribute translate :: Stmt;
        translate = ifThenElse(cond, body, skip()); }

seq: s::Stmt ::= s1::Stmt s2::Stmt      { ... }

ifThenElse: s::Stmt ::= cond::Expr   thenS::Stmt   elseS::Stmt      { ... }

skip: s::Stmt ::=      { ... }
```

Rewrites rules extracted from the grammar:

- $while(cond, body) \Longrightarrow ifThen(cond, seq(body, while(cond, body)))$
- $doWhile(body, cond) \Longrightarrow seq(body, while(cond, body))$ and
- $ifThen(cond, body) \Longrightarrow ifThenElse(cond, body, skip())$

Fig. 1. A higher-order attribute grammar fragment for which attribute evaluation will terminate for the *doWhile* and *ifThen* constructs, but may not for the *while* construct. The rewrite rules generated from this grammar are shown at the bottom.

implementing domain-specific languages. In 1989 Vogt et al. [2] introduced higher-order attribute grammars (HOAGs) in which syntax trees can be computed and passed around as attribute values.

Vogt et al. originally defined a *well-defined* higher-order attribute grammar to be one that (*i*) is *complete*,[1] so that every synthesized, inherited and local attribute instance on a node has a defining equation; (*ii*) is *non-circular*, so that no attribute value depends on itself, directly or indirectly; and (*iii*) creates a *finite number of new syntax trees* during attribute evaluation. For the first two conditions, Vogt et al. specify extended versions of Knuth's tests for completeness[2] and non-circularity. Unlike in attribute grammars without higher-order attributes, attribute evaluation may fail to terminate since an unbounded number of new trees may be created; evaluating attributes on a tree may result in new trees being created and attributes being evaluated on them. This explains why Vogt et al. included the third condition in their definition of well-definedness. Taken together, these analyses prevent abnormal termination of attribute evaluation. That said, the term "well-defined" is typically used to describe attribute grammars that meet only these first two criteria: completeness and non-circularity. Vogt's later Ph.D. dissertation adopts this alternative use of the term [10]. This is the definition used in this paper.

An example of a well-defined attribute grammar that may create an unbounded number of attributed syntax trees is shown in Fig. 1. These *attributed* syntax trees are held in higher-order local attributes (originally called non-terminal attributes), where they can be given values for their inherited attributes and thus their synthesized attributes can be computed. Syntax trees held in higher-order synthesized or inherited attributes cannot be provided with inherited attributes and are thus considered to be unattributed trees. Here a higher-order local attribute *translate* of type *Stmt* is computed. It may be supplied with inherited attributes and queried for synthesized attributes, though these are not shown in the figure. Each evaluation of the *translate* attribute on the *while*-loop production may create another tree containing a *while*-loop, and thus an unbounded number of such trees may be created. Note that the *translate* attribute on the *doWhile*-loop does not, by itself, result in a nonterminating evaluation of attributes. It is the *while*-loop translation that is the cause of nontermination. Any analysis to detect the creation of an unbounded number of trees will be conservative; to see this consider the conditional expression in the *while*-loop example. An analysis to guarantee that no evaluation of attributes will create infinitely many attributed trees, combined with a well-definedness tests (completeness and non-circularity tests), would be sufficient to ensure higher-order attribution termination. While Vogt et al. describe a condition required to ensure non-termination (see Section 8 on related work for details), it does not seem to have been implemented or evaluated. The analysis presented here is the first, to our knowledge, that uses the structure of the equations and not just the attribute dependencies imposed by them in an analysis for termination of tree creation.

This paper, which extends our previous work [11], fills in this gap with a conservative analysis to ensure that during attribute evaluation, tree creation terminates. This analysis uses rewrite rules to model tree construction and production

[1] This use of "complete" applies to attribute grammars and should not be confused with the use of "complete" in proof-theoretic settings where soundness and completeness are considered. Both uses appear in this paper but the meaning is made clear from context.

[2] Knuth [1, p. 132] did not use the term "completeness" to define a property of an attribute grammar but instead incorporated the notion of completeness into its definition.