SAT-solving Linear Ordered Attribute Grammars

L. Thomas van Binsbergen[1], Jeroen Bransen[2], and Koen Claessen[3]

[1] todo, `todo@todo.co.uk`
[2] Utrecht University, Utrecht, The Netherlands, `J.Bransen@uu.nl`
[3] todo, `koen@chalmers.se`

Abstract. A nice abstract here.

Keywords: Attribute Grammars, SAT-Solving

1 Introduction

2 Linear Ordered Attribute Grammars

An *Attribute Grammar* is a context-free grammar with inherited and synthesized attributes assigned to every non-terminal and a graph that represents dependencies between those attributes at the production level.

Definition 1. *Attribute Grammar*
An Attribute Grammar (AG) is a triple $\langle G, A, D\rangle$, where:

- *Context-free grammar $G = \langle N, T, P, S\rangle$ contains a set of non-terminals N, a set of terminals T, a set of production rules P and a start symbol S. Every $p \in P$ is of the form $X_{p,0} \to X_{p,1} \dots X_{p,|p|}$, with $lhs(p) = X_{p,0}$ and $rhs(p) = X_{p,1}, \dots, X_{p,|p|}$, where each $X_{p,i} \in \{lhs(p)\} \cup rhs(p)$ is called a field of p and an occurrence of some non-terminal $X \in N$, i.e. $\exists(X \in N)\ (X_{p,i} = X)$.*
- *A set $A(X) = A_{inh}(X) \cup A_{syn}(X)$ is defined for all $X \in N$. From the attributes we infer the attribute occurrences gathered in the set $A_P(p) = A_{in}(p) \cup A_{out}(p)$, where $A_{in}(p)$ and $A_{out}(p)$ are the input and output occurrences of p respectively.*

$$\begin{aligned} A_{in}(p) = &\{\ X_{p,0} \cdot a \ \mid\ X \cdot a \in A_{inh}(lhs(p))\ \} \\ &\cup \{\ X_{p,i} \cdot a \ \mid\ X_{p,i} \in rhs(p),\ X = X_{p,i},\ X \cdot a \in A_{syn}(X))\ \} \end{aligned}$$

$$\begin{aligned} A_{out}(p) = &\{\ X_{p,0} \cdot a \ \mid\ X \cdot a \in A_{syn}(lhs(p))\ \} \\ &\cup \{\ X_{p,i} \cdot a \ \mid\ X_{p,i} \in rhs(p),\ X = X_{p,i},\ X \cdot a \in A_{inh}(X))\ \} \end{aligned}$$

- *A dependency graph $D(p)$ indicates that attribute a is used in the semantic function definition of attribute b when $(a \to b) \in D(p)$.*

The definition of AGs given above is not a complete definition in the sense that it does not contain enough information to generate executable code from it - the actual semantic function definitions are missing, for example. However it contains all the information we need to define the problem of finding a static evaluation order for all Linear Ordered Attribute Grammars and deciding whether an AG is an LOAG.

Definition 2. *Linear Ordered Attribute Grammars*

A Linear Ordered Attribute Grammar (LOAG) is an AG that satisfies the following properties:

- *For all $X \in N$ there exists a graph $R_X(x)$ that satisfies:*
  - Totality*: There must be an edge between every inherited and synthesized pair:*
  $$\forall(i \in A_{inh}(X), s \in A_{syn}(X))$$
  $$(i \to s) \in R_X(X) \vee (s \to i) \in R_X(X)$$
- *For all $p \in P$ there exists a graph $R_P(p)$ that satisfies:*
  - Feasibility*: The graph must include the dependencies, i.e.*
  $$D(p) \subseteq R_P(p).$$
  - Consistency*: The graph must be consistent with R_X, i.e.*
  $$\forall(X \in \{lhs(p)\} \cup rhs(p), X = X_{p,i})$$
  $$((X \cdot a \to X \cdot b) \in R_X(X) \Rightarrow (X_{p,i} \cdot a \to X_{p,i} \cdot b))$$
  - Orderability*: The graph $R_P(p)$ must be acyclic.*

Graph R_P serves the same purpose as graph ED_P from Kastens and from $R_X(X)$ we can infer Kastens' interfaces[1].

2.1 Satisfiability

Using the above definition for LOAGs we can define a Boolean Satisfiability Problem that determines whether an arbitrary AG is an LOAG.

Firstly, be defining a variable $x_{i,s}$ for every element of the Cartesian product $i \times s$, with $i \in A_{inh}(X)$ and $s \in A_{syn}(X)$ for all $X \in N$. Secondly, by saying $x_{a,b} = \top$ when $a = (X_{p,i} \cdot a')$, $b = (X_{p,i} \cdot b')$ and $(X_{p,i} \cdot a' \to X_{p,i} \cdot b') \in D_P(p)$ (*feasibility*). Thirdly, by assigning every variable $\top$ or $\perp$ (*totality*) under the constraints that the graph R_P the assignments imply is cycle free (*orderability*). Note that *consistency* is guaranteed by making sure that a variable represents the edge between attributes (non-terminal level) as well as the edges between all the occurrences of that edge (production level).

References

1. Kastens, U.: Ordered attributed grammars. Acta Informatica **13**(3) (1980) 229–256