Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 322 (2016) 119–134

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Formal Attributes Traceability in Modular Language Development Frameworks

Walter Cazzola[a,1] Paola Giannini[b,2] Albert Shaqiri[a,1]

[a] *Università degli Studi di Milano, Dipartimento di Informatica*

[b] *Università del Piemonte Orientale, Computer Science Institute, Disit*

Abstract

Modularization and component reuse are concepts that can speed up the design and implementation of domain specific languages. Several modular development frameworks have been developed that rely on attributes to share information among components. Unfortunately, modularization also fosters development in isolation and attributes could be undefined or used inconsistently due to a lack of coordination. This work presents 1) a type system that permits to trace attributes and statically validate the composition against attributes lack or misuse and 2) a correct and complete type inference algorithm for this type system. The type system and inference are based on the Neverlang development framework but it is also discussed how it can be used with different frameworks.

Keywords: modularity and composition, modular language implementation, formal validation of the composition, type inference

1 Introduction

Domain specific languages (DSLs) are getting more and more relevant nowadays but their development is still difficult and this contains their proper spread. One way to ease DSL development, consists in maximizing reuse by modularizing the language and its implementation in the composition of loosely coupled *language component*s where a language component is any language-oriented concept that should be part of the language shipped together with its implementation. According to this trend several modular development frameworks have been developed such as Lisa [9], JastAdd [5], Silver [11], Spoofax [7] and Neverlang [10].

The implementation of each language component provides a syntactic description of the language concept itself and the code necessary to support the expected

⋆ Partly funded by "Progetto MIUR PRIN CINA Prot. 2010LHT4KM".

[1] Email: {cazzola,shaqiri}@di.unimi.it

[2] Email: giannini@di.unipmn.it

http://dx.doi.org/10.1016/j.entcs.2016.03.009

semantics for such a concept. Composition is typically driven by the syntactic description of the language component that provides an interface to the other language components. Even if loosely coupled the code realizing the semantic of the different language components relies on data computed by the other components and the sharing of these data is typically delegated to and relies on the presence of attributes [8]. Being the composition syntax-driven, semantic constraints as attributes presence are rarely considered at development/composition time.

Modular development eases the reuse of language components fostering their separate development. Language components are developed in isolation and reused as black boxes relying on naming conventions or similar expedients. This may become unreliable when the language components can be separately compiled and dynamically composed, as in Neverlang, since they rely on information that is not part of component's interface. Without some static check, the composition may turn out in a real mess.

Apart from Spoofax [7] that provides a language transformation engine all the other approaches exploit variants of the attribute grammars [8] and syntax direct translation [1]. A typical (dangerous) situation consists of a language component's implementation that relies on an attribute that should be provided by the implementation of another component and the attribute is not defined, is defined with a different name or is inconsistently used. In such a situation, even if the composition could be done and the compiler generated, it will fail when it is used. Also when the attributes are declared as in Lisa [9] or precomputed as in Silver [11] there is still no assurance that they are consistently used.

In this work, we formalize the problem of attributes traceability over separate language component implementations by providing a type system that statically validates the composition with respect to the attributes. The validation is tailored on the Neverlang framework but it can be easily adapted to other modular language development frameworks as Lisa and Silver.

The paper is organized as follows. Section 2 introduces the problem of well-definedness in attribute grammars, how this is contextualized to the framework Neverlang and an overview of the proposed solution. Section 3 introduces the formalization of Neverlang slice that is used in Section 4 to define a small-step operational semantics that specifies how the semantic actions define, access and modify the attributes of syntax-trees. Section 5 introduces a type system for a type decorated version of slices which prevent runtime errors and states the soundness result. Section 6 outlines the type inference algorithm and states that it is correct and complete for the type system. In Sect. 7 some related work and the application of the presented system to them is discussed. In Sect. 8 we draw some conclusions.

2 Overview

Well-definedness in attribute grammars. The problem of ensuring that a grammar is well-defined has been addressed since the genesis of attribute grammars. In the context of pure attribute grammars, the *well-definedness* is traditionally