FUNCTIONAL EVALUATION OF
STRONGLY NON CIRCULAR TYPOL SPECIFICATIONS

Isabelle ATTALI
INRIA – Sophia-Antipolis
Route des Lucioles
06565 Valbonne Cedex, France
e-mail: ia@trinidad.inria.fr

Jacques CHAZARAIN
CNRS I3S & Math. Dpt
University of Nice
06034 Nice Cedex, France
e-mail: jmch@cerisi.cerisi.fr

Abstract

Typol is a language to implement Natural Semantics inside the Centaur Programming Environment. We explain why Natural Semantics can be considered as a generalization of the Attribute Grammars formalism. This relationship suggests introducing in the Typol setting some concepts from the Attribute Grammars framework. With these concepts, we can provide specialized evaluators to some classes of Typol programs, instead of the general Prolog machine currently used to execute such programs. More precisely, we prove that under acceptable conditions, the unification process is no longer required and can be replaced by a simple pattern-matching mechanism. This makes it possible to give a functional implementation of Typol programs.

1 Introduction

The description of semantic properties in the natural deduction style advocated by G. Plotkin [20], has been used by Damas and Milner [6] and developed by G. Kahn in his group inside the Centaur project [9,16,10]. We must also indicate the seminal work of Warren [21].

A semantic specification is represented by a set of inference rules of the form

\[ H_1 \vdash T_1 : S_1 \quad \cdots \quad H_n \vdash T_n : S_n \]

which constitutes, together with type information, a Typol program. To compute the semantic value, say \( S_0 \), of an abstract syntax term \( T_0 \), given some initial environment \( H_0 \), means to prove the goal sequent \( H_0 \vdash T_0 : S_0 \) in this deductive system.

In this paper, we study the structure of Natural Semantics specifications in order to associate functional evaluators to some classes of Typol programs. The natural deduction formalism can be considered, in a sense, as a generalization of Attribute Grammars (AG). More precisely, we explain how to translate a given AG specification into a Typol program. The generated Typol programs verify some specific properties, such as:

- the proof tree of a goal is always isomorphic to the abstract syntax tree of the term \( T_0 \);
- there is no link nor constraint within input attributes because they are composed of pairwise distinct variables;
- there is one rule for each abstract syntax operator.
This explains why, in [2,1], some restrictive hypotheses, due to the AG framework, were necessary. In other words, since the usual domains of AG and Logic Programming are different (values vs terms), the mandatory hypotheses concerned the absence of links and constraints within input attributes. In the present work, these unnatural restrictions are simply released because proofs are entirely established in the logical setting of Natural Semantics; here, Attribute Grammars are only used as a guideline to suggest new concepts but not as a tool for proofs. This makes it possible to obtain new results under natural hypotheses.

We adapt for the Typol setting some classical notions from AG: input/output attributes, minimal argument selector, Strongly Non Circular programs. We prove the feasibility of evaluating Typol programs in successive passes. Our major result is the definition of a rather large class of Typol programs in which the unification process can be replaced by a less costly pattern-matching process (Theorems 1 & 2). This generalizes the results of [2,1] to a larger class of programs and is closely related to [19] in spirit, but not in scope (Prolog evaluation vs functional evaluation). Furthermore, we give, under acceptable hypotheses, a functional implementation (instead of the usual Prolog machine) for Typol programs evaluation.

This paper belongs to a recent field of investigations: relationship between Attribute Grammars and Logic Programming [7,14].

Section 2 formally defines, with a number of illustrative examples, the Natural Semantics formalism and outlines a comparison with Attribute Grammars. Section 3 gives an operational semantics to the Typol formalism, defines various notions of circularities, and namely the Strongly Non Circularity, describes a reduction technique useful to transform a circular program (requiring an unification) into separate Strongly Non Circular programs. In Section 4, we present our theoretical results about the unification-free execution of non trivial classes of Typol programs. Finally, in Section 5 we discuss implementation issues for a functional evaluator of Typol programs based on the proof technique of the theorems. Section 6 concludes the paper.

2 Some definitions about the Natural Semantics formalism

A Typol program is essentially a collection of inference rules. Each inference rule is composed of a finite set of premises (which is empty for an axiom) and a conclusion. Premises and conclusion of a rule are relations represented by sequents in the Gentzen natural deduction style [12].

The object languages are manipulated via their abstract syntax, defining a many-sorted algebra. This presentation allows quite compact and readable specifications of semantic properties such as static semantics, dynamic semantics, and translations. Moreover, it is an executable formalism because each rule can be translated into an Horn clause.

In order to illustrate our definitions and results, we use well-known examples of Typol programs such as the dynamic semantics of a small Pascal-like language, the static and dynamic semantics of Mini-ML [4], and a translator from regular expressions to deterministic automata [3,2].

In this section, we describe more formally the notions of abstract syntax, sequents, rules, and we compare the Typol formalism with the Attribute Grammars one.