Semantical Evaluations as Monadic Second-Order Compatible Structure Transformations

Bruno Courcelle

LaBRI (CNRS, UMR 5800),
Université Bordeaux-I, France **

Abstract. A transformation of structures τ is *monadic second-order compatible* (*MS-compatible*) if every monadic second-order property P can be effectively rewritten into a monadic second-order property Q such that, for every structure S, if T is the transformed structure $\tau(S)$, then $P(T)$ holds iff $Q(S)$ holds.
We will review *Monadic Second-order definable transductions* (*MS -transductions*) : they are MS-compatible transformations of a particular form, *i.e.*, defined by monadic second-order (MS) formulas.
The unfolding of a directed graph into a tree is an MS-compatible transformation that is not an MS- transduction.
The MS-compatibility of various transformations of semantical interest follows. We will present three main cases and discuss applications and open problems.

Overview of the lecture

Our working logical language is *Monadic Second-Order Logic*, *i.e.*, the extension of First-Order Logic with variables denoting sets of elements of the considered structures. It enjoys a number of interesting properties regarding decidability and construction of polynomial algorithms [4].

We consider certain semantical evaluations that can be formalized as transformations of discrete structures like graphs or trees (and not as mappings from terms to values belonging to semantical domains as this is usual in denotational semantics).

Our main concern will be to identify transformations such that the verification of an MS property P, of a structure $T = \tau(S)$ reduces to the verification of an MS property Q of the input structure S, where Q depends only on τ and P and, of course, of the fixed relational signatures of S and T.

** 351 cours de la Libération, F-33405 Talence
email: courcell@labri.fr,
WWW: http://dept-info.labri.fr/~courcell/ActSci.html

In such a case, if the MS theory of an infinite structure S is decidable (which means that there exists an algorithm that decides whether a monadic second-order formula is true or not in S), then so is that of $T = \tau(S)$. We say that τ is *Monadic Second-order compatible* (*MS-compatible*).

Monadic Second-order definable transductions (*MS-transductions* in short) have been surveyed in [2]. The idea is that $T = \tau(S)$ if T is defined by MS formulas inside the strucure formed of k disjoint copies of S (where k and these MS formulas are fixed and constitute the logical specification of τ). That an MS-transduction is MS-compatible is pretty clear for those familiar with the notion of interpretation in model theory.

An obvious consequence of the definition is that the size of the domain of $T = \tau(S)$ is at most k times the size of that of S. In particular the *unfolding* operation which transforms a finite graph into an infinite tree is not an MS-transduction. However, the unfolding operation is MS-compatible [3,6].

Let us consider some examples and their semantical motivations.

Example1:

The structure S is a finite or infinite *transition system*, *i.e.*, a directed labelled graph, given with a special vertex called the *initial state*. The *unfolding* of S from the initial state is a tree (usually infinite), the tree of finite paths starting from the initial state, that represents the behaviour of the transition system.

Example 2:

S is here a finite or infinite *directed acyclic graph* representing a finite or infinite term with *shared* subterms. Labels attached to vertices and edges make unambigous the description of such a term by a graph. *Unsharing*, the operation that reconstructs the term, is a special case of unfolding.

As an example of such a graph, we can take $f \rightrightarrows f \rightrightarrows f \rightrightarrows a$ with f a binary function symbol and a a constant.

It unshares into the term $f(f(f(a,a),f(a,a)),f(f(a,a),f(a,a)))$.

By looking at sizes, one can see that unsharing is not an MS-transduction.

Example 3:

S is here a *recursive applicative program scheme*, as those considered in [1], and T is the infinite term called an *algebraic tree*. It is the infinite term resulting from a certain form of unfolding, involving term substitutions. Here is an example, consisting of a single equation:

$\varphi(x) = f(x, \varphi(g(x))$

This scheme unfolds into the algebraic tree:

$f(x, f(g(x), f(g(g(x)), f(g(g(g(x))), ...))))$.

The scheme is actually not given by a graph, but a graph representation fitting our formal framework is easy to build. The transformation from graphs representing schemes (consisting of several mutually recursive equations) with function symbols of bounded arity to algebraic trees is MS-compatible. It follows in particular that the MS theory of an algebraic tree is decidable [3,5].

Example 4:

Hyperalgebraic trees are defined as algebraic trees except that the "unknown functions" in schemes may take parameters of function type. Such schemes have been first investigated by W. Damm [7] and more recently by Knapik et al. [8,9].

Example 5:
We introduce new symbols to represent first-order substitutions. It is proved in [5] that the mapping from terms to terms that evaluates substitutions, i.e., that eliminates substitution symbols is MS-compatible.

Here is an example. $Sub_{x,y}$ denotes the ternary function such that $Sub_{x,y}(t, t_1, t_2)$ evaluates into the term $t[t_1/x, t_2/y]$, i.e., the result of the simultaneous substitution of t_1 for x and t_2 for y in t.

For instance, the term

$Sub_{x,y}(\ Sub_{x,u}(f(x,x,x,u),a,y),b,c)$

evaluates to $Sub_{x,y}(f(a,a,a,y),b,c)$ and then to $f(a,a,a,c)$. (Note that b disappears because the variable x has no occurrence in the term $f(a,a,a,c)$.

A central result is the MS-compatibility of the unfolding mentioned in Examples 1 and 2. The case of Example 5 is proved in [5] and the results of Examples 3 and 4 follow.

Actually, the main result behind all this is the MS-compatibility of a transformation of structures defined by Muchnik (improving a definition by Shelah and Stupp), which builds a kind of tree $Tree(S)$ over any structure S. (If S is a set of two elements, then $Tree(S)$ is the infinite complete binary tree). This result is proved in [10]. It subsumes the case of unfolding and we think it may help to solve some open questions in [8,9].

We will discuss applications and open problems related to Example 5.

References

1. B. Courcelle: Recursive Applicative Program Schemes, chapitre 9 of *Handbook of Theoretical Computer Science, Volume B*, J. Van Leeuwen ed., Elsevier 1990, pp. 459-492.

2. B. Courcelle: Monadic second order graph transductions: A survey. *Theoretical Computer Science*, **126**, 53–75,1994

3. B. Courcelle: The Monadic Second-Order Logic of Graphs, IX: Machines and their behaviours, , *Theoret. Comput. Sci.* **151** (1995) 125-162.

4. B. Courcelle: The expression of graph properties and graph transformations in monadic second-order logic, Chapter 5 of the "Handbook of graph grammars and computing by graph transformations, Vol. 1 : Foundations", G. Rozenberg ed., World Scientific (New-Jersey, London), 1997, pp. 313-400.

5. B. Courcelle, T. Knapik: The evaluation of first-order substitution is monadic second-order compatible, 2001, *Theoret. Comput. Sci.* to appear.

6. B. Courcelle, I. Walukiewicz: Monadic second-order logic, graph coverings and unfoldings of transition systems, *Annals of Pure and Applied Logic*, **92** (1998) 35-62.

7. W. Damm, The IO- and OI hierarchies, *Theoret. Comput. Sci.* **20** (1982) 95-208.

8. T. Knapik, D. Niwinski, P. Urzyczyn : Deciding monadic theories of hyperalgebraic trees**,** in *Typed lambda-calculi and applications*, Lec. Notes Comp. Sci., **2044** (2001) 253-267.

9. T. Knapik, D. Niwinski, P. Urzyczyn : Higher-order pushdown trees are easy**,** FOSSACS 2002, this volume.

10. I. Walukiewicz : Monadic second-order logic on tree-like structures. Proceedings STACS'96, Lec. Notes Comp. Sci. **1046** (1996) 401-414. (Full paper to appear).

Full texts of submitted or unpublished papers and other references can be found from: http://dept-info.labri.u-bordeaux.fr/~courcell/ActSci.html.