ELSEVIER

Theoretical Computer Science 281 (2002) 177–206

Theoretical Computer Science

www.elsevier.com/locate/tcs

The evaluation of first-order substitution is monadic second-order compatible

Bruno Courcelle[a,*], Teodor Knapik[b]

[a] *LaBRI (CNRS laboratory 5800), Université Bordeaux 1, 351 Cours de la Libération, 33405 Talence Cedex, France*

[b] *ERMIT, Université de la Réunion, BP 7151, 97715 Saint Denis Messageries, Cedex 9, La Réunion, France*

Abstract

We denote first-order substitutions of finite and infinite terms by function symbols indexed by the sequences of first-order variables to which substitutions are made. We consider the *evaluation* mapping from infinite terms to infinite terms that evaluates these substitution operations. This mapping may perform infinitely many nested substitutions, so that a term which has the structure of an infinite string can be transformed into one isomorphic to an infinite binary tree. We prove that this mapping is *monadic second-order compatible* which means that a monadic second-order formula expressing a property of the output term produced by the evaluation mapping can be translated into a monadic second-order formula expressing this property over the input term. This implies that, deciding the monadic second-order theory of the output term reduces to deciding that of the input term. As an application, we obtain another proof that the monadic second-order properties of the *algebraic trees*, which represent the behaviours of recursive applicative program schemes, are decidable. This proof extends to *hyperalgebraic trees*. These infinite trees correspond to certain recursive program schemes with functional parameters of arbitrary high type.

Keywords: Monadic second-order logic; First-order substitution; Algebraic tree; Hyperalgebraic tree; Decidability of logical theories

* Corresponding author.

E-mail addresses: courcell@labri.fr (B. Courcelle), knapik@univ-reunion.fr (T. Knapik).

URLs: http://www.labri.fr/∼courcell, http://www.univ-reunion.fr/∼knapik

0304-3975/02/$ - see front matter
PII: S0304-3975(02)00012-9

1. Introduction

For investigating the recursive applicative program schemes considered in the 1970s by Nivat and others [11,12,17,18] (see [5] for a survey), it is useful to associate with every such a scheme an infinite term that represents its computations in all semantical domains. Useful informations about a scheme can be obtained from the corresponding tree and used, for instance, for rewriting the scheme in a simpler or in a standard way [17]. It is thus useful to know which properties of these terms (called *algebraic trees* in the above quoted works) are decidable. A fundamental theorem by Rabin [20] (see [21]) states that the properties expressible in monadic second-order logic (MS logic in short) are decidable over regular trees, a proper subclass of that of algebraic trees. Monadic second-order logic is quite powerful (it subsumes temporal logics) but regular trees are insufficient to represent computations of recursive applicative program schemes. However, the monadic second-order properties of algebraic trees are decidable [8]. We give a new proof of this result which, furthermore, extends to the *hyperalgebraic* trees that correspond to certain higher-level program schemes, with parameters of function and functional types, already considered by Damm [15].

A first result about the decidability of the MS theory of hyperalgebraic trees is given in [19] where, so called *safe* applicative schemes of level 2 are considered. In these schemes procedure names carry functional parameters in addition to individual ones but, according to the safety restriction, a parameter of the basic type should not occur within a functional argument of a procedure call. In an essential way, the proof of [19] uses paths in λ-graphs. The latter are borrowed form the geometry of interactions [1] and the theory of optimal reductions [2].

The present paper is independent of these λ-calculus related theories. We develop several tools for the MS logic of higher-level trees. Our first tool is a signature of graph operations studied by Courcelle, Olariu, Engelfriet and others [10,16,6,9,13]. We use operations which allow to add edges to a vertex labelled graph, from every vertex labelled by, say, a to every vertex labelled by, say, b. Other operations rename labels and a binary operation builds the disjoint union of two labelled graphs. Using these operations, graphs are generated from constants, each of which represents a single labelled vertex. In such a way, a graph may be represented by a finite or infinite term over the corresponding signature of graph operations. The set of such terms is organized into a continuous algebra and the unique continuous morphism from this algebra into the algebra of graphs is called *the value morphism*. The graph represented by the term is the image of the term under the value morphism. Our first result shows that this value morphism is a *monadic second-order transduction* (an *MS transduction* in short). MS transductions are transformations of logical structures specified in MS logic; see [7,9]. This implies that the value morphism preserves the decidability of the MS theory of the structures to which it is applied. Hence the MS theory of a graph defined by an infinite regular term is decidable.

Our second main tool is an explicit operation of first-order substitution of terms. This operation specifies the variables to which substitutions are made. The arguments are thus: the term on which the substitution acts and the terms which are substituted. We show that the evaluation of substitutions is an MS-compatible operation.

A transformation of structures f is *MS-compatible* if for every MS formula φ, one can effectively construct an MS formula ψ such that for every structure S, S satisfies ψ iff $f(S)$ satisfies φ. Every MS-transduction is MS compatible, but some MS-compatible mappings, like the *unfolding* of a graph into a tree (by a result of [14], see Theorem 2.11), are not MS transductions.

Our proof uses in a crucial way the MS-compatibility of unfolding.

We also consider second-order substitutions in the context of recursive applicative program schemes. By reducing second-order substitutions to first-order ones, we reduce the functionality of the types of the variables in recursive definitions. In such a way, we establish that every algebraic tree may be obtained from a regular tree with first-order substitution operators by evaluating these substitution operators (a result known from [15]). This gives another proof of the decidability of the MS theory of every algebraic tree.

Since the evaluation of first-order substitutions is MS-compatible, by evaluating them on algebraic trees, we get more complex trees with decidable MS theories. This process may be iterated. We obtain that each tree in the strict hierarchy defined by Damm in [15] has a decidable MS theory. It is not clear yet whether these trees represent all *lambda*-schemes, i.e., all systems of typed mutually recursive definitions using *lambda*-abstraction in the right-hand sides of equations.

2. Graphs, terms, monadic second-order logic: basic definitions

In Sections 2.1–2.5, we review from previous articles definitions and notation concerning relational structures and MS logic, finite and infinite terms, graphs and graph operations. In Section 2.5 we prove that the value mapping associating a graph to an infinite term is an MS-transduction, hence is MS-compatible. In Section 2.6 we introduce notation for new graph operations derived from those of Section 2.5. In Section 2.7 we review graph unfolding.

2.1. Structures and monadic second-order logic

We let R be a finite set of relation symbols, each of them, say r, being given with an *arity* $\rho(r)$ in $\mathbf{N}_+$. We denote by $\mathscr{S}(R)$ the set of finite or countable *R-structures*, i.e., of tuples of the form $S=\langle D_S,\ (r_S)_{r\in R}\rangle$ where $r_S\subseteq D_S^{\rho(r)}$ for $r\in R$. For two structures S and S' in $\mathscr{S}(R)$, we write $S\subseteq S'$ (read S is *included in* S') if $D_S\subseteq D_{S'}$, and $r_S\subseteq r_{S'}$ for each r in R.

We recall that *monadic second-order logic* (MS logic for short) is first-order logic augmented with (uppercase) variables denoting subsets of the domain of the considered structure, and new atomic formulas of the form $x\in X$ expressing the membership of x in a set X. We denote by $MS(R,\mathscr{X})$ the set of MS formulas over R with free variables in $\mathscr{X}$ (the set of all individual and set variables.)

A property of structures (or of elements and/or of sets of elements of a structure) is MS-*definable* if it can be expressed by an MS formula.

We denote by $S \equiv S'$ the existence of an isomorphism between two structures S and S'. If $L, L' \subseteq \mathscr{S}(R)$, we write $L \equiv L'$ if every structure of L is isomorphic to a structure in L' and vice-versa.

2.1.1. MS-transductions

A *transduction* of structures is a multivalued mapping: $\mathscr{S}(R) \to \mathscr{S}(R')$, formally handled as a mapping $f : \mathscr{S}(R) \to \wp(\mathscr{S}(R'))$ such that $S \equiv S'$ implies $f(S) \equiv f(S')$, where $\wp(-)$ denotes a powerset operation. We say that f as above is *MS-compatible* [14] if there exists a total recursive mapping $f^{\#} : MS(R', \emptyset) \to MS(R, \emptyset)$ such that $S \models f^{\#}(\varphi)$ iff $S' \models \varphi$ for some $S' \in f(S)$. We call $f^{\#}(\varphi)$ the *backwards translation* of φ relative to f.

Let $L \subseteq \mathscr{S}(R)$. We say that a transduction $f : \mathscr{S}(R) \to \mathscr{S}(R')$ is *deterministic on* L if for every $S \in L$, $f(S)$ is not empty and its elements are pairwise isomorphic.

We now consider transductions defined by MS formulas. *A parameterless MS-definable* transduction $f : \mathscr{S}(R) \to \mathscr{S}(R')$ is defined as follows, from $k \in \mathbf{N}_+$ and MS formulas α in $MS(R, \emptyset), \delta_1, \ldots, \delta_k$ in $MS(R, \{x\}), \theta_{r, i_1, \ldots, i_n}$ in $MS(R, \{x_1, \ldots, x_n\})$, for $r \in R', n = \rho(r), 1 \leqslant i_1, \ldots, i_n \leqslant k$:

(1) $f(S)$ is non-empty iff $S \models \alpha$.
(2) Assuming $S \models \alpha$, then $f(S) = \{T\}$ where T is constructed as follows:
   - $D_T = D_1 \times \{1\} \cup \cdots \cup D_k \times \{k\} \subseteq D_S \times \{1, \ldots, k\}$;
   - each D_i is $\{x \in D_S \mid S \models \delta_i(x)\}$;
   - each relation r_S, for $r \in R'$ is defined on T as the union of the sets of tuples of the form $\{((x_1, i_1), \ldots, (x_n, i_n)) \mid S \models \theta_{r, i_1, \ldots, i_n}(x_1, \ldots, x_n)\}$, for all $i_1, \ldots, i_n \in \{1, \ldots, k\}$.

The tuple $\langle \alpha, \delta_1, \ldots, \delta_k, (\theta_{r, i_1, \ldots, i_{\rho(r)}})_{r \in R, 1 \leqslant i_1, \ldots, i_{\rho(r)} \leqslant k} \rangle$ is called a *definition scheme*.

An MS-definable transduction is *k-copying*, if $k > 1$ (and *noncopying* if $k = 1$). In all cases, f is deterministic: $L \to \mathscr{S}(R')$ where $L = \{S \in \mathscr{S}(R) \mid S \models \alpha\}$ and its domain is of course MS-definable.

A parameterless MS-definable transduction f is MS-compatible [7,9]. One can even define a backwards translation $f^{\#}(\varphi)$ for MS formulas φ over R' with free variables. If φ has p free variables, then $f^{\#}(\varphi)$ has kp free variables. We refer the reader to [7,9] for details.

2.1.2. MS-transductions with parameters

We now extend the previous definitions in order to define (by MS formulas) transductions: $\mathscr{S}(R) \to \mathscr{S}(R')$ that are not deterministic. Let $p_1, \ldots, p_n$ be n unary relation symbols, that we will call *parameters* ($p_1, \ldots, p_n \notin R \cup R'$). We let $\Pi_R : \mathscr{S}(R \cup \{p_1, \ldots, p_n\}) \to \mathscr{S}(R)$ be the mapping that "forgets" the relations $p_1, \ldots, p_n$. (It is actually a noncopying MS-definable transduction.)

A transduction $f : \mathscr{S}(R) \to \mathscr{S}(R')$ is *MS-definable with parameters* (we also say that it is an *MS-transduction*) if there exists an MS-definable subset L of $\mathscr{S}(R \cup \{p_1, \ldots, p_n\})$ and a parameterless MS-definable transduction $g : L \to \mathscr{S}(R')$ such that:

$$f(S) = \{g(S') \mid S' \in L, \Pi_R(S') = S\}.$$

It is *k-copying* or *noncopying* if g is so. The set $\{S\in\mathscr{S}(R)\mid f(S)\neq\emptyset\}$ is MS-definable: it is defined by the formula $\exists X_1,\ldots,X_n.\alpha[X_1/p_1,\ldots,X_n/p_n]$ where $\alpha\in MS(R\cup\{p_1,\ldots,p_n\},\emptyset)$ defines L and X_i/p_i denotes the substitution of X_i for p_i in α. (We replace $p_i(x)$ by $x\in X_i$ for every i and x.).

An MS-transduction f as above is MS-compatible: for every $\varphi\in MS(R',\emptyset)$ one takes $f^{\#}(\varphi)$ equal to

$$\exists X_1,\ldots,X_n.(\alpha\wedge g^{\#}(\varphi))[X_1/p_1,\ldots,X_n/p_n].$$

If $f:\mathscr{S}(R)\to\mathscr{S}(R')$ and $g:\mathscr{S}(R')\to\mathscr{S}(R'')$ are two MS-transductions then the transduction $(g\circ f):\mathscr{S}(R)\to\mathscr{S}(R'')$ defined by

$$(g\circ f)(S)=\bigcup\{g(S')\mid S'\in f(S),S\in\mathscr{S}(R)\}$$

is an MS-transduction. It is noncopying if f and g are so. See [7,9].

We now consider as an example and for later use the mapping $S\mapsto S/\approx$ which associates with a structure S its quotient by an MS-definable equivalence relation $\approx$. We will see that this mapping is a deterministic noncopying MS-transduction, and its definition uses one parameter.

Let $S=\langle D_S,(r_S)_{r\in R}\rangle$ be a structure in $\mathscr{S}(R)$, and $\approx$ be an equivalence relation on D_S. We denote by $[d]$ the equivalence class of $d\in D_S$. We let $S'=S/\approx$ be the structure in $\mathscr{S}(R)$ defined as follows:

$$D_{S'}=D_S/\approx$$

$$r_{S'}([d_1],\ldots,[d_n]) \text{ holds iff } r_S(d'_1,\ldots,d'_n) \text{ holds for some } d'_1\approx d_1,\ldots,d'_n\approx d_n.$$

We call it the *quotient structure of* S by $\approx$.

Lemma 2.1. *Let η be a formula in $MS(R,\{x,y\})$ such that, for every $S\in\mathscr{S}(R)$ the relation $\{(x,y)\in D_S^2\mid S\models\eta(x,y)\}$ is an equivalence relation $\approx$. The mapping $S\mapsto S/\approx$ is a deterministic noncopying MS-transduction.*

Proof. We let p be a parameter. We let $\alpha\in MS(R\cup\{p\})$ be such that $S\models\alpha$ iff every equivalence class of $\approx$ contains one and only one element of p_S.

The set p_S can be taken as domain for the structure $S/\approx$. Hence there exists a parameterless noncopying MS-transduction $g:L\to\mathscr{S}(R)$ such that $L=\{S'\in\mathscr{S}(R\cup\{p\})\mid S'\models\alpha\}$, and $g(S')=\Pi_R(S')/\approx$ for $S'\in L$, where $\approx$ is the equivalence relation on $\Pi_R(S')$ defined by η. It follows that the mapping $S\mapsto S/\approx$ is an MS-transduction.

For every S there are several possible sets p_S that satisfy α, hence several structures S' such that $\Pi_R(S')=S$. But the structures $g(S')$ are all isomorphic. Although the transduction $S\mapsto S/\approx$ is deterministic, we need a parameter for defining it as an MS-transduction. □

2.2. Finite and infinite terms

We review definitions and notation from [3,11,12]. We let F be a finite set of function symbols, each of them, f, given with an arity $\rho(f)\in\mathbf{N}$. We let X be a finite

set of variables. We denote by $T(F,X)$ (resp. $T^{\infty}(F,X)$) the set of finite (resp. finite or infinite) well-formed terms written over $F \cup X$. We let $k_F = \mathrm{Max}\{\rho(f) \mid f \in F\}$ and $F_k = \{f \in F \mid \rho(f) = k\}$.

The occurrences in $t \in T^{\infty}(F,X)$ of the symbols of $F \cup X$ can be formalized as Dewey words, i.e. as elements of $\{1,\ldots,k_F\}^*$. A *tree-domain* is a subset L of $\{1,\ldots,k_F\}^*$ such that:

(i) L is prefix-closed,

(ii) if $ui \in L$ with $u \in \{1,\ldots,k_F\}^*$ and $j \in \{1,\ldots,k_F\}, 1 \leqslant j < i$ then $uj \in L$.

A term $t \in T^{\infty}(F,X)$ can be formalized by a mapping $Sym_t : Dom(t) \to F \cup X$ where $Dom(t) \subseteq \{1,\ldots,k_F\}^*$ is a tree-domain and for each $u \in Dom(t)$, $Sym_t(u)$ is the symbol *occurring at u in t*. This mapping is subject to the following condition: for all $u \in Dom(t)$ and $i \in \{1,\ldots,k_F\}$, we have $ui \in Dom(t)$ iff $i \leqslant \rho(Sym_t(u))$. (For a variable $x \in X$, we let $\rho(x) = 0$.)

Finite terms will be, as usual, denoted by finite words. For an example the term $t = f(x, f(x, g(a)))$ has the corresponding mapping Sym_t that associates f with ε and 2, x with 1 and 21, g with 22 and a with 221.

Two terms t and t' are equal iff $Dom(t) = Dom(t')$ and $Sym_t = Sym_{t'}$. A term t is finite iff $Dom(t)$ is finite. If $t \in T^{\infty}(F,X)$ and $u \in Dom(t)$, we denote by t/u *the subterm of t issued from u*. It is defined by:

$$Dom(t/u) = \{v \in \{1,\ldots,k_F\}^* \mid uv \in Dom(t)\},$$

$$Sym_{t/u}(v) = Sym_t(uv) \quad \text{for } v \in Dom(t/u).$$

We now recall that $T^{\infty}(F,X)$ is a complete partial order. We assume that F contains a special nullary symbol denoted by Ω. We define a partial order on $T^{\infty}(F,X)$ by letting:

$$t \prec t' \text{ iff } Dom(t) \subseteq Dom(t') \text{ and for every } u \in Dom(t),$$

$$\text{either } Sym_t(u) = \Omega \text{ or } Sym_t(u) = Sym_{t'}(u).$$

Every increasing sequence of terms $t_0 \prec t_1 \prec t_2 \prec \cdots \prec t_n \prec \cdots$ has a least upper bound in $T^{\infty}(F,X)$ denoted by $t = \sup_{n \geqslant 0}(t_n)$. (We have $Dom(t) = \bigcup_{n \geqslant 0} Dom(t_n)$.) Finally, every infinite term $t \in T^{\infty}(F,X)$ is the least upper bound of an increasing sequence of finite terms $(t^{(n)})_{n \geqslant 0}$ defined as follows:

$$Dom(t^{(n)}) = \{u \in Dom(t) \mid |u| \leqslant n\},$$

$$Sym_{t^{(n)}}(u) = \begin{cases} Sym_t(u) & \text{if } |u| < n, \\ \Omega & \text{if } |u| = n. \end{cases}$$

(Hence $t^{(0)} = \Omega$.)

We will use this fact in order to extend to infinite terms a k-ary mapping, say $m : T(F,X)^k \to D$ where D is a complete partial order and m is monotone, by letting:

$$\hat{m}(t_1,\ldots,t_k) = \sup_{n \geqslant 0}(m(t_1^{(n)},\ldots,t_k^{(n)})).$$

In this way we define an ω-continuous mapping $\hat{m}: T^{\infty}(F,X)^k \to D$ that extends m ("ω-continuous" means "monotone and continuous over infinite increasing sequences", sometimes also called "ω-chains"; see [3,4,5,11]).

2.3. Graphs

All graphs will be directed and at most countable. An edge e of a graph G has a *source* $src_G(e)$, a target $tgt_G(e)$ and a label in a finite set of edge labels, usually denoted by A. All graphs will be *simple* which means that no two edges have same source, same target and same label.

We denote by V_G the set of vertices of a graph G and by E_G its set of edges. (An element of E_G is a triple in $V_G \times A \times V_G$). Furthermore, vertices may have labels from a finite set, say P. A vertex may have several labels or none. Hence, a graph G can be identified with the relational structure:

$$\langle V_G, (edg_{aG})_{a\in A}, (lab_{pG})_{p\in P}\rangle,$$

where lab_{pG} is the set of vertices labelled by p, $edg_{aG} = \{(src_G(e), tgt_G(e)) \mid e \text{ is an edge labelled by } a\}$.We denote by $\mathcal{G}(A,P)$ the class of graphs with A and P as respective sets of edge and vertex labels.

A *walk* in G from x to y is a sequence of edges $(e_1,\ldots,e_n)$ such that $tgt_G(e_i) = src_G(e_{i+1})$ for every $i=1,\ldots,n-1$, $x = src_G(e_1)$ and $y = tgt_G(e_n)$. A walk as above is a *path* if $tgt_G(e_i) \neq src_G(e_{j+1})$ for $1 \leqslant i < j \leqslant n-1$. A *circuit* is a path from x to x for some x.

If $B \subseteq A$, we say that an edge is a *B-edge* if its label is in B. A *B-walk*, a *B-path*, a *B-circuit* is a walk, a path or a circuit, all edges of which are B-edges. If $G \in \mathcal{G}(A,P)$, and $u \in V_G$, we let $Acc(G,u)$ be the set of vertices of G that are the end of a walk beginning at u. We let $G^{Acc}(u)$ denote the subgraph of G induced by the set of vertices $Acc(G,u)$.

Let $G,H \in \mathcal{G}(A,P)$. We say that G is a subgraph of H, if $G \subseteq H$ where G and H are considered as structures (see Section 2.1). We say that G is an *induced subgraph* of H, written $G \subseteq_i H$, if $G \subseteq H$ and $lab_{pG} = lab_{pH} \cap V_G$ and $edg_{aG} = edg_{aH} \cap (V_G \times V_G)$ for all p and a.

A *homomorphism* $h: G \to H$ is a mapping $V_G \to V_H$ such that:

(i) $h(lab_{pG}) \subseteq lab_{pH}$ for each $p \in P$;

(ii) $((h(x),h(y)) \in edg_{aH}$ if $(x,y) \in edg_{aG}, a \in A$.

It is an isomorphism iff it is bijective, we have $=$ instead of $\subseteq$ in condition (i) and we have "iff" instead of "if" in condition (ii).

If $G \in \mathcal{G}(A,P)$ and $\sim$ is an equivalence relation on V_G, then the quotient graph $G/\sim$ is defined as a quotient structure (see Section 2.1). In particular we have a canonical surjective homomorphism: $G \to G/\sim$.

For $B \subseteq A$ we denote by $Contr_B$ the mapping from $\mathcal{G}(A,P)$ to $\mathcal{G}(A-B;P)$ that contracts all B-edges and removes the resulting loops. Formally, $Contr_B(G) = H$ where H is the graph $G/\sim$ with all its B-edges deleted, and $\sim$ is the equivalence relation on V_G generated by the union of the relations edg_{bG} for $b \in B$.

2.4. Terms and rooted graphs

A *rooted* graph is a graph G (directed as are all our graphs) given with a distinguished vertex called the *root* from which every vertex is accessible by a directed path. We denote by $root_G$ the root of G. Since graphs may have circuits, several vertices may be chosen as the root.

We will use a vertex label rt to distinguish the root ($lab_{rtG}=\{root_G\}$). We denote by $\mathscr{D}(A,P)$ the set of rooted graphs with set of edge labels A and set of vertex labels P. We denote by $\mathscr{D}'(A,P)$ the set of those such that lab_{rtG} is singleton, but do not satisfy necessarily the accessibility condition. Hence $\mathscr{D}(A,P)\subseteq\mathscr{D}'(A,P)\subseteq\mathscr{G}(A,P\cup\{rt\})$.

A graph $G\in\mathscr{D}(A,P)$ is *a tree* iff it has no circuit and has at most one path between any two vertices.

We will handle terms in $T^{\infty}(F,X)$ as trees as well as mappings on tree-domains. Let t belong to $T^{\infty}(F,X)$ with $Dom(t)$, and Sym_t as in Section 2.2. We let $G(t)$ be the tree in $\mathscr{D}(\{1,\ldots,k_F\};F\cup X)$ defined by

$$G=\langle Dom(t),(edg_{iG})_{1\leqslant i\leqslant k_F},(lab_{pG})_{p\in P}\rangle,$$

where $P=(F|\{\Omega\})\cup X\cup\{rt\}$, and:

$$edg_{iG}=\{(u,ui)\,|\,u,ui\in Dom(t)\},1\leqslant i\leqslant k_F,$$

$$lab_{pG}=\{u\,|\,Sym_t(u)=p\},\quad \text{if } p\in F\cup X,$$

$$lab_{rtG}=\{\varepsilon\}.$$

Note that $G(\Omega)$ is a single vertex labelled just by rt. In the graph $G(f(\Omega,\Omega))$ the vertices corresponding to the two occurrences of Ω have no label.

From a structure G isomorphic to $G(t)$ for some $t\in T^{\infty}(F,X)$, one can determine t in a unique way, as one can check easily. Hence, in order to prove that $t_1,t_2\in T^{\infty}(F,X)$ are equal, it is enough to prove that the structures $G(t_1)$ and $G(t_2)$ are isomorphic.

Let $t\in T^{\infty}(F,X)$ and $u\in Dom(t),u\neq\varepsilon$. Let $G(t)^{Acc}(u)$ be the subgraph of $G(t)$ induced by $Acc(G(t),u)$ according to the definition given in Section 2.3. Let $H=G(t)^{Acc}_{rt}(u)$ be this graph augmented with a "root label" i.e., we let $lab_{rtH}=\{u\}$. Then it is not hard to see that H is isomorphic to $G(t/u)$. The isomorphism $h:G(t/u)\to H$ is given by $h(w)=uw$ (we recall that the sets of vertices of H and $G(t/u)$ are subsets of $\{1,\ldots,k_F\}^*$).

Remark 2.2. If $t_1\prec t_2$ then $G(t_1)\subseteq G(t_2)$.

For expressing properties of terms t in MS logic, we will use the corresponding structures $G(t)$. In particular, the relation *w is an ancestor of u* which is exactly $w\leqslant u$, (w is a prefix of u) is expressible in MS logic in $G(t)$ because the transitive closure of an MS definable relation is MS-definable (see [9, Lemma 1.7]).

2.5. Operations on graphs

We will use operations on graphs which are very close to those used in [13] for multilabelled graphs (see pp. 96–97 of that article).

We fix A and P as in Section 2.3. The first operation is *disjoint union* which is binary. For $G,H\in\mathcal{G}(A,P)$ we denote by $G\oplus H$ their disjoint union. If necessary, we replace H by a copy disjoint with G and one defines $K=G\oplus H$ as follows:

$$V_K = V_G \cup V_H \quad (\text{with } V_H \cap V_G = \emptyset),$$

$$edg_{aK} = edg_{aG} \cup edg_{aH} \quad \text{for } a\in A,$$

$$lab_{pK} = lab_{pG} \cup lab_{pH} \quad \text{for } p\in P.$$

The next operation is unary. It adds edges as follows. For $p,q\in P, a\in A, G\in\mathcal{G}(A,P)$ we let $H=add_{p,q,a}(G)$ be such that

$$V_H = V_G,$$

$$lab_{rH} = lab_{rG} \quad \text{for all } r\in P,$$

$$edg_{aH} = edg_{aG} \cup (lab_{pG} \times lab_{qG}),$$

$$edg_{bH} = edg_{bG} \quad \text{if } b\in A, b\neq a.$$

The third operation, also unary, modifies vertex labels as follows. For $p\in P, Q\subseteq P$, $G\in\mathcal{G}(A,P)$, we let $H=ren_{p\to Q}(G)$ be defined by

$$V_H = V_G,$$

$$edg_{aH} = edg_{aG} \quad \text{for all } a\in A,$$

$$lab_{rH} = lab_{rG} \cup lab_{pG} \quad \text{if } r\neq p, r\in Q,$$

$$lab_{rH} = lab_{rG} \quad \text{if } r\neq p, r\notin Q,$$

$$lab_{pH} = \emptyset \quad \text{if } p\notin Q,$$

$$lab_{pH} = lab_{pG} \quad \text{if } p\in Q.$$

In words, this means that each label p of a vertex of G is replaced by the set of labels Q. This set Q may be empty; it may contain p.

For each $p\in P$ we let $\mathbf{p}$ be a nullary symbol denoting the graph with one vertex labelled by p and no edge. Finally, we also introduce a nullary symbol Ω denoting the empty graph $\emptyset$. (It will be useful for dealing with infinite terms denoting countable graphs.)

We let $\mathcal{F}_{A,P}$ denote this set of operations and constant symbols. Every term t in $T(\mathcal{F}_{A,P})$ defines a finite graph in $\mathcal{G}(A,P)$ that we will denote by $val(t)$. Because of the need to take disjoint copies in the evaluation of $G\oplus H$, the graph $val(t)$ is only defined up to isomorphism. This makes complicated to designate precise vertices of $val(t)$. In

order to overcome this difficulty we present an alternative, more precise construction of $val(t)$, where the set $V_{val(t)}$ is fixed in a unique way.

Construction 2.3. *We let $t \in T(\mathscr{F}_{A,P})$ and we construct from it a graph G as follows. The graph G is intended to be isomorphic to $val(t)$. We let V_G be the set of "leaves" of t labelled by an element of P (as opposed to by Ω). Hence*

$$V_G = \{u \mid u \in Dom(t), Sym_t(u) \in P\}. \tag{2.1}$$

Our next task is to determine the set $LAB_t(u) = \{q \in P \mid q \text{ labels } u \text{ in } val(t)\}$ for $u \in V_G$.

Let R be the set of operations of the form $ren_{p \to Q}$ for $p \in P, Q \subseteq P$. We first consider the special case of $t \in T(R \cup P)$, i.e. of $t \in R^*P$ (it is convenient to identify $r_1(r_2(\cdots r_n(\mathbf{p})\cdots)) \in T(R \cup P)$ with the word $r_1 r_2 \ldots r_n p$). The graph $val(t)$ has then a unique vertex say s. (Because of the renaming operations r_i, we need not have p in $LAB_t(s)$.) The sets of labels of the unique vertex of $val(p), val(r_n p), \ldots, val(r_1 r_2 \ldots r_n p)$ can be computed by a finite automaton reading $r_1 r_2 \ldots r_n p$ from right to left. The set of the states of such automaton is the powerset $\wp(P)$, $\emptyset$ is the initial state, Q is the final state and the transitions are of the form $P' \xrightarrow{ren_{p' \to Q'}} (P' \setminus \{p'\}) \cup Q'$ or $\emptyset \xrightarrow{p'} \{p'\}$. Hence for every $Q \subseteq P$ the set of words $r_1 \ldots r_n p \in R^*P$ such that $LAB_{r_1 \ldots r_n p}(s) = Q$ is a regular language. It is written L_Q.

We now go back to the general case where $t \in T(\mathscr{F}_{A,P})$ and $u \in Dom(t)$. We let $u = u_1 u_2 \ldots u_n$ (with $u_i \in \{1, \ldots, k_F\}$) and $SYM_t(u)$ be the word in $(\mathscr{F}_{A,P})^*$ defined as

$$Sym_t(\varepsilon) Sym_t(u_1) Sym_t(u_1 u_2) \ldots Sym_t(u).$$

(Hence $SYM_t(u)$ is the sequence of operations seen on the path from the root to u in the tree representing t.) Then, if u is a leaf of t having a label in P, we have:

$$LAB_t(u) = Q \subseteq P \quad \text{iff} \quad y \in L_Q, \tag{2.2}$$

where y is the word in R^*P obtained from $SYM_t(u)$ by removing all symbols not in R. Removing from $SYM_t(u)$ the symbols not in R corresponds to the fact that only the operations from R modify the labelling of vertices.

We now determine the existence of edges between two vertices of the graph G (intended to be $val(t)$), given as elements of $Dom(t)$.

We extend as follows the function SYM_t defined above. If $u \in V_G$ and $w \in Dom(t)$, $w \leqslant u$ (i.e., w is "above u in t"), we let $u = wv_1 v_2 \ldots v_n$ (with $v_1, \ldots, v_n \in \{1, \ldots, k_F\}$, $n \leqslant 1$), and $SYM_t(w, u)$ be the word:

$$Sym_t(w) Sym_t(wv_1) Sym_t(wv_1 v_2) \ldots Sym_t(wv_1 \ldots v_n) \in (\mathscr{F}_{A,P})^*.$$

Hence $SYM_t(u) = SYM_t(\varepsilon, u)$.

Let us consider $u, u' \in V_G$. We put in G an edge from u to u' labelled by a iff there is in t an occurrence w of the operation $add_{p,q,a}$ and $p \in LAB_{t/w1}(v), q \in LAB_{t/w1}(v')$ where $u = w1v, u' = w1v'$. Intuitively, these conditions mean that in the subgraph of G defined by $t/w1$, the vertex u has label p, the vertex u' has label q, so that $add_{p,q,a}$ introduces an edge from u to u' labelled by a; this edge remains in G. (Since $add_{p,q,a}$ is unary,

1 *follows* w *in both* u *and* u'.) *We denote by* $C(t,w,u,u',p,q,a)$ *this condition. Hence, in* G *we have, if* $u,u' \in V_G$, $a \in A$: $(u,u') \in edge_{aG}$ *iff*:

$$\text{there exist } w \in Dom(t) \text{ with } w \leqslant u, w \leqslant u',\ Sym_t(w) = add_{p,q,a}$$

$$\text{for some } p,q \in P, \text{ such that } C(t,w,u,u',p,q,a) \text{ holds.} \tag{2.3}$$

Condition $C(t,w,u,u',p,q,a)$ *is written as follows*:

There exist $Q,Q' \subseteq P$ *such that* $p \in Q$, $q \in Q'$, $y \in L_Q$ *and* $y' \in L_{Q'}$,

where y, y' *are the words in* R^*P *obtained from* $SYM_t(w,u)$ *and* $SYM_t(w,u')$

by removing the symbols not in R.

This ends Construction 2.3.

We claim that the graph G defined in this way from t is isomorphic to $val(t)$. This follows actually from the observations we made along with the definition.

From now on and unless specified otherwise $val(t)$ will denote the graph defined from t as explained above. We have in particular:

Lemma 2.4. *If* $t,t' \in T(\mathscr{F}_{A,P})$ *and* $t \prec t'$ *then* $val(t)$ *is an induced subgraph of* $val(t')$.

Proof. Every occurrence in t of a symbol $p \in P$ is also one in t'. Hence $V_{val(t)} \subseteq V_{val(t')}$.

The labelling of u in $V_{val(t)}$ and the edges from u to u' for $u,u' \in V_{val(t)}$ depend only on the operation symbols in $SYM_t(w)$ for the elements w of $Dom(t)$ such that $w \leqslant u$ or $w \leqslant u'$. And $SYM_{t'}(w) = SYM_t(w)$ for these w. Hence $val(t) \subseteq_i val(t')$. □

If $t \in T^\infty(\mathscr{F}_{A,P})$ and $t = \sup_{n \geqslant 0}(t_n)$ where t_n is an increasing sequence of finite terms, then we have an increasing sequence of induced subgraphs:

$$val(t_0) \subseteq_i val(t_1) \subseteq_i \cdots \subseteq_i val(t_n) \subseteq_i \cdots$$

and we define $val(t)$ as its union. Note that for each $n \in \mathbf{N}$, $V_{val(t_n)} \subseteq Dom(t) \subseteq \{1,\ldots,k_F\}^*$. Hence we take the countable union of an increasing sequence of graphs with sets of vertices all included in $Dom(t)$. It is not hard to verify that if $t = \sup_{n \geqslant 0}(t'_n)$ where t'_n is another increasing sequence of finite terms, then the union of the graphs $val(t'_n)$ is the same. Hence $val(t)$ is well-defined. Furthermore, Construction 2.3 works for t infinite as well as finite.

We have defined val as a mapping from $T^\infty(\mathscr{F}_{A,P})$ to $\mathscr{G}(A,P)$. Since a term in $T^\infty(\mathscr{F}_{A,P})$ is defined in a unique way from the rooted graph $G(t)$, we can extend the notation by letting $val(G(t)) = val(t)$.

Proposition 2.5. *The mapping that associates* $val(t)$ *with* $G(t)$ *for* $t \in T^\infty(\mathscr{F}_{A,P})$ *is an MS-transduction.*

Proof. It is straightforward to translate Construction 2.3 into an MS-transduction. By (2.1), $V_{val(G(t))}$ is the set of vertices u of $G(t)$ such that $lab_{pG(t)}(u)$ holds for some $p \in P$.

An MS-formula $\psi_Q(u)$ can be constructed that translates condition (2.2) of Construction 2.3. This is possible by the following two observations: first one can define in $G(t)$ the sequence of ancestors of a given u (simply by using the relations $edg_{iG(t)}$) so that we can define the word $SYM_t(u)$; the second observation is that finite automata on words can be translated into MS formulas [21]. Having thus the formulas $\psi_Q(u)$ for $Q\subseteq P$, we let $\lambda_p(u)$ defined as $\bigvee_{p\in Q\subseteq P}\psi_Q(u)$. It expresses that u has label p in $val(t)$. For the same reasons, condition (2.3) of Construction 2.3 can be translated into an MS formula $\theta_a(u,u')$ expressing the existence of an a-edge in $val(t)$ from u to u'. Hence we obtain a noncopying parameterless definition scheme:

$$\langle \alpha, \varphi(u), (\theta_a(u,u'))_{a\in A}, (\lambda_p(u))_{p\in P}\rangle,$$

where $\varphi(u)$ is $\bigvee_{p\in P} lab_p(u)$ and α is an MS formula (easy to construct) expressing that a given structure is of the form $G(t)$ for some $t\in T^\infty(\mathscr{F}_{A,P})$. This concludes the proof. □

2.6. Interpretations

We have defined on $\mathscr{G}(A,P)$ a structure of $\mathscr{F}_{A,P}$-algebra. If F is a set of function symbols and for each $f\in F$ of arity k we define a term $\mathscr{I}(f)$ in $T(\mathscr{F}_{A,P'},\{x_1,\ldots,x_k\})$ (where P' is a finite superset of P) intended to define a k-ary total function on $\mathscr{G}(A,P)$, then we make $\mathscr{G}(A,P)$ into an F-algebra $\mathscr{G}^{\mathscr{I}}(A,P)$. Each operation of it is called a *derived operation* [3,4] of the $\mathscr{F}_{A,P}$-algebra $\mathscr{G}(A,P)$.

In order to simplify the notation, we will not distinguish between the term $\mathscr{I}(f)$ in $T(\mathscr{F}_{A,P},\{x_1,\ldots,x_k\})$ and the total k-ary function on $\mathscr{G}(A,P)$ it defines, namely the function which maps each tuple $(G_1,\ldots,G_k)$ of $\mathscr{G}(A,P)^k$ to the graph $val(\mathscr{I}(f))(G_1,\ldots,G_k)$. We will furthermore use $\mathscr{I}$ to define for every $t\in T^\infty(F)$ its *interpretation* $\mathscr{I}nt(t)\in\mathscr{G}(A,P)$. We let $\mathscr{I}nt(t):T(F)\to\mathscr{G}(A,P)$ be defined in a natural way by

$$\mathscr{I}nt(f)=\mathscr{I}(f) \quad \text{if } f\in F_0,$$

$$\mathscr{I}(\Omega)=\emptyset \quad \text{(the empty graph)},$$

$$\mathscr{I}nt(f(t_1,\ldots,t_k))=\mathscr{I}(f)(\mathscr{I}nt(t_1),\ldots,\mathscr{I}nt(t_k)),$$

$$\text{for } f\in F_k, k\geqslant 1 \text{ and } t_1,\ldots,t_k\in T(F).$$

Note that $\mathscr{I}nt(t)=val(t')$ where t' is obtained from t by substituting $\mathscr{I}(f)$ for each $f\in F$. This substitution, called *second-order substitution*, will be recalled from [3] in Section 4.2. Since the operations of $\mathscr{F}_{A,P}$ are monotone for graph inclusion we have

Lemma 2.6. *If $t,t'\in T(F)$ and $t\prec t'$ then $\mathscr{I}nt(t)\subseteq_i\mathscr{I}nt(t')$.*

Hence we can define $\mathscr{I}nt(t)$ for t infinite in $T^\infty(F)$ as the union of the graphs $\mathscr{I}nt(t_n)$ which form an increasing sequence for inclusion, where $(t_n)_{n\geqslant 0}$ is an increasing sequence of finite terms such that $t=\sup_n(t_n)$.

Lemma 2.7. *The mapping* $t \mapsto \mathcal{I}nt(t)$ is an MS-transduction from $T^{\infty}(F)$ *into* $\mathcal{G}(A,P)$.

Proof. This mapping is the composition of two MS-transductions. The first one transforms t into a term over $\mathcal{F}_{A,P}$, and the second one is the mapping *val* of Proposition 2.5. □

2.7. Coverings and unfoldings

We recall the unfolding operation considered in [14] as well as the notion of covering. From now on, A,P are fixed finite sets of edge and vertex labels.

If $G \in \mathcal{G}(A;P)$, $x \in V_G$, we denote by $Suc_{aG}(x)$ the set $\{y \in V_G \mid (x,y) \in edg_{aG}\}$. Since graphs are simple, this set is in bijection with the set of a-edges with source x.

Definition 2.8. Let $G,H \in \mathcal{D}(A;P)$. A homomorphism $h: G \to H$ is *a covering*, if

(i) it is surjective,
(ii) $h(lab_{pG}) = lab_{pH}$ for all $p \in P$,
(iii) $h(root_G) = root_H$,
(iv) for every $a \in A$ and $x \in V_G$, h is a bijection of $Suc_{aG}(x)$ onto $Suc_{aH}(h(x))$.

We also say that G is *a covering of* H.

If $h: G \to H$ is a covering and π is a walk in H from x to y and $h(x') = x$, then there exists $y' \in V_G$ and a walk π' in G from x' to y' the image of which by h is π. The proof is easy by induction on the length of π.

Proposition 2.9. *Every graph* $H \in \mathcal{D}(A;P)$ *has a covering* G *that is a tree.*

Proof. We let V_G be the set of finite walks in H the origin of which is $root_H$. We put in V_G the empty walk ε from the root to itself; it has no edge.

We let $root_G = \varepsilon$ and $edg_{aG} = \{(\pi,\pi') \in V_G^2 \mid \pi'$ is π extended by one $\{a\}$-edge$\}$. We let $h: G \to H$ map $\pi \in V_G$ to x, where π goes from $root_H$ to x. Hence $h(\varepsilon) = root_H$ and $lab_{pG} = h^{-1}(lab_{pH})$ for each p.

It is straightforward to verify that G is a tree and that $h: G \to H$ is a covering. □

This tree is denoted by $Un(H)$, and, as in [14], we call it the *unfolding* of H (Fig. 1).

Proposition 2.10. *Let* $T,G,H \in \mathcal{D}(A;P)$ *such that* T *is a tree,* $t: T \to H$ *and* $g: G \to H$ *are coverings. There exists a unique homomorphism* $k: T \to G$ *such that* $g \circ k = t$. *This homomorphism is a covering. If* G *is a tree, it is an isomorphism.*

Proof. For every n we let T_n be the subtree of T induced by the set of vertices at distance at most n from the root. We prove by induction on n that for every n there is a unique homomorphism $k_n: T_n \to G$ such that $g(k_n(x)) = t(x)$ for all $x \in V_{T_n}$.

The case of $n = 0$ is clear since T_0 reduces to $root_T$.

Let us assume that k_n is known, and that y is in T_{n+1} but not in T_n. We have an edge $x \to y$ in T, labelled by, say, $a \in A$ where $x \in T_n$. Since t is a bijection of $Suc_{aT}(x)$

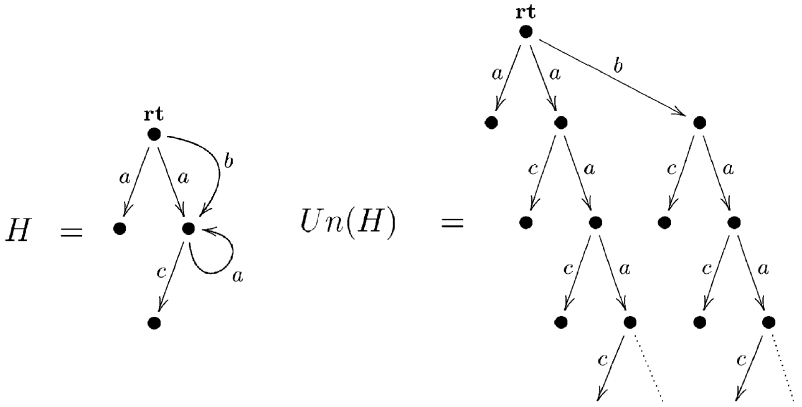


Fig. 1. A graph and its unfolding

onto $Suc_{aH}(t(x))$ and g is one of $Suc_{aG}(k_n(x))$ onto $Suc_{aH}(t(x))$ we can define $k_{n+1}(y)$ as $g^{-1}(t(y))$ for $y \in Suc_{aT}(x)$. Of course we take $k_{n+1}(z) = k_n(z)$ for every $z \in V_{T_n}$. Hence we have the desired homomorphism $k_{n+1} : T_{n+1} \to G$. For k, we take the common extension of all the homomorphisms k_n, $n \geqslant 0$.

At each level n, we have no choice for defining k_n since we want $g \circ k_n$ to coincide with t on T_n. Hence k is the unique homomorphism $T \to G$ such that $g \circ k = t$. Moreover, it is clear from the construction that it is a covering.

If $G, G' \in \mathscr{D}(A;P)$, and $m : G' \to G$ is a covering where G is a tree, then G' is a tree (otherwise G would have circuits or distinct paths between a same pair of vertices) and m is a bijection (for the same reason) and, moreover, an isomorphism. By applying this remark to T, we get that k is an isomorphism: $T \to G$. It follows that, up to isomorphism, $Un(H)$ is the only covering of H that is a tree. □

We extend the mapping Un to graphs in $\mathscr{D}'(A;P)$ as follows. We let

$$Un(H) = Un(H_{rt}^{Acc}(root_H)),$$

i.e., we apply Un to the subgraph of H induced by the vertices accessible from the root.

Theorem 22 of [14] can be reformulated as follows with the notation of the present paper:

Theorem 2.11. *For all finite sets A and P, the mapping Un from $\mathscr{D}'(A,P)$ to $\mathscr{D}(A,P)$ is MS-compatible.*

3. The evaluation of first-order substitutions

In this section, we establish our main theorem, saying that the evaluation of first-order substitutions is an MS-compatible mapping.

3.1. First-order substitutions

As in Section 2.2 we let F and X be finite sets of function symbols and of variables, respectively. For each n-tuple $\bar{x}=(x_1,\ldots,x_n)$ of pairwise distinct variables in X, we introduce an $(n+1)$-ary operation $sub_{\bar{x}}$ such that $sub_{\bar{x}}(t_0,t_1,\ldots,t_n)$ is the result of the simultaneous *substitution* of t_i for x_i in t_0 for all $i=1,\ldots,n$. A common notation for substitution used in [3] is $t_0[t_1/x_1,\ldots,t_n/x_n]$. This operation is the textual substitution for finite terms $t_0,t_1,\ldots,t_n$ (represented by words, say in Polish prefix notation) and is extended by continuity to infinite terms (because it is monotone in all its arguments [3, Proposition 3.3.3]). Other characterizations are given in [3].

If $t\in T^{\infty}(F,X)$, we let $Var(t)=\{x\in X \mid Sym_t(u)=x \text{ for some } u\in Dom(t)\}$. We have:

$$Var(sub_{\bar{x}}(t_0,t_1,\ldots,t_n)) = (Var(t_0)-\{x_1,\ldots,x_n\}) \cup \bigcup\{Var(t_i) \mid x_i\in Var(t_0), 1\leqslant i\leqslant n\}.$$

We let $Sub(X)$ be the set of operation symbols $sub_{\bar{x}}$ where $\bar{x}$ is a nonempty sequence of pairwise distinct variables in X and $\rho(sub_{\bar{x}})=|\bar{x}|+1$. Every term $t\in T(F\cup Sub(X),X)$ can be evaluated into a term $Eval(t)\in T(F,X)$ just by performing the substitutions as prescribed by the operations $sub_{\bar{x}}$. For $t=f(t_1,\ldots,t_k)$, $f\in F$, $\rho(f)=k$, $t_i\in T(F\cup Sub(X),X)$ we have of course:

$$Eval(f(t_1,\ldots,t_k)) = f(Eval(t_1),\ldots,Eval(t_k))$$

(hence functions in F are not evaluated).

Since $sub_{\bar{x}}: T(F,X)^{|\bar{x}|+1}\to T(F,X)$ is monotone, the mapping $Eval: T(F\cup Sub(X),X)\to T(F,X)$ is monotone and extends into an ω-continuous mapping $T^{\infty}(F\cup Sub(X),X)\to T^{\infty}(F,X)$ by $Eval(t)=\sup_n(Eval(t_n))$ where $(t_n)_{n\geqslant 0}$ is an increasing sequence of finite terms with least upper bound t. (We recall that "ω-continuous" means "monotone and continuous over infinite increasing sequences".)

Example 3.1. Let t be the infinite term over symbols f (binary), x and sub_x defined by

$$t = sub_x(f(x,x), sub_x(f(x,x),\ldots.)\ldots)$$

i.e., the infinite term such that:

$$t = sub_x(f(x,x),t).$$

Then $T=Eval(t)$ satisfies

$$T = f(T,T)$$

hence is isomorphic to the complete infinite binary tree.

Remark 3.2. We have $sub_{\bar{x}}(\Omega,t_1,\ldots,t_n)=\Omega$.

It follows that $Eval(t)=\Omega$ if $t=sub_{\bar{x}}(sub_{\bar{y}}(sub_{\bar{z}}(\cdots),\cdots)$ where t, considered as an infinite tree, has a left-most branch with only symbols in $Sub(X)$. This corresponds to

the fact that a recursive definition like $w = w[f(x)/x]$ where $w \in T^\infty(F,X)$ defines the "bottom" element, here Ω in the case of terms. See also the last example in Section 4.2.

Our next objective is to prove that the mapping $Eval : T^\infty(F \cup Sub(X),X) \to T^\infty(F,X)$ is MS-compatible, where a term t is represented by the tree $G(t)$ as explained in Section 2.4.

3.2. Evaluating substitutions

We denote by $\mathscr{D}_1(A,B;P)$ the set of graphs in $\mathscr{D}(A;P)$ such that every vertex has at most one outgoing B-edge. We denote by $\mathscr{D}'_1(A,B;P)$, the corresponding subsets of $\mathscr{D}'(A;P)$ for which we waive the accessibility condition (see Section 2.4).

The main theorem of this section is the following:

Theorem 3.3. *Let F,X be finite.*

(1) *One can define two finite sets A,P where $\varepsilon \in A$, and an interpretation $\mathscr{I}nt : T^\infty(F \cup Sub(X),X) \to \mathscr{D}_1(A,\{\varepsilon\};P)$ such that for every $t \in T^\infty(F \cup Sub(X),X)$ we have*:

$$G(Eval(t)) = Contr_\varepsilon(Un(\mathscr{I}nt(t))).$$

(2) *The mapping $Eval : T^\infty(F \cup Sub(X),X) \to T^\infty(F,X)$ is MS-compatible.*

Proof. We first prove (2) assuming (1).

The mappings $\mathscr{I}nt$ and $Contr_\varepsilon$ are MS-transductions (by Lemmas 2.7 and 2.1) hence are MS-compatible. The mapping Un is MS-compatible according to Theorem 2.11. Hence $Eval$ is MS-compatible.

We now start the proof of (1). We first define A and P as follows:

$$A = \{1,\dots,k_F\} \cup \{\varepsilon\},$$

$$P = F \cup X \cup P_{aux} \cup \{rt\}.$$

where P_{aux} is the set

$$\{aux_i \mid 1 \leqslant i \leqslant Card(X)\} \cup \{rt_i \mid 0 \leqslant i \leqslant Max\{k_F, Card(X)\}\}$$

of auxiliary "new" labels. In order to make easily understandable our construction, we show immediately an example.

Example 3.4. If $t = sub_{x,y,z}(t_0,t_1,t_2,t_3)$ where

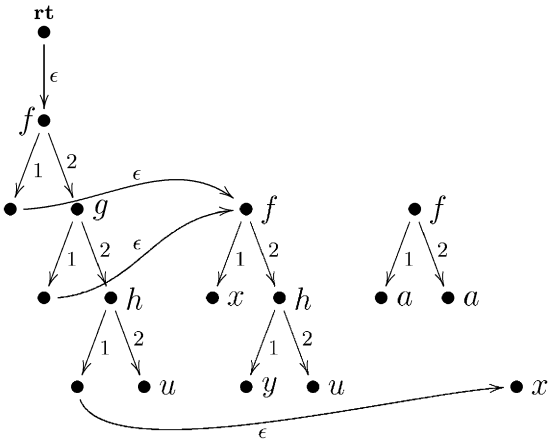
$$t_0 = f(x,g(x,h(z,u))),$$

$$t_1 = f(x,h(y,u)),$$

$$t_2 = f(a,a),$$

$$t_3 = x,$$

then $\mathscr{I}nt(t)$ is the following graph:



By *Un* the subgraph $G(t_1)$ is duplicated (because x has two occurrences in t_0), and $G(t_2)$ disappears (because it is not linked to $G(t)$, since y has no occurrence in t_0).

It is clear that $Contr_{\varepsilon}(Un(G))$ is the graph $G(Eval(t))$ and

$$Eval(t) = f(f(x,h(y,u)),g(f(x,h(y,u)),h(x,u))).$$

In order to shorten the definition of $\mathscr{I}nt$ we introduce an auxiliary operation built from the basic ones. If $q_1,\ldots,q_m \in P$, we let, for any graph G,

$$rem_{q_1,\ldots,q_m}(G) = ren_{q_1\to\emptyset}(ren_{q_2\to\emptyset}(\ldots(ren_{q_m\to\emptyset}(G))\cdots)).$$

This operation removes all labels q_i.

We are now ready to specify $\mathscr{I}nt : T(F\cup Sub(X),X)\to\mathscr{G}(A,P)$. We will see later that the object graphs are actually in $\mathscr{D}'_1(A,\{\varepsilon\};P)$.

Definition 3.5. We define $\mathscr{I}$:

(i) $\mathscr{I}(\Omega)=\mathbf{rt}$ (we do not set $\mathscr{I}(\Omega)=\emptyset$ as in Section 2.6)

(ii) $\mathscr{I}(w)=ren_{rt\to\{w,rt\}}(\mathbf{rt})$ if $w\in X\cup F_0$,

(iii) For $f\in F_k$ we let

$$\begin{aligned}&\mathscr{I}(f)(G_1,\ldots,G_k)\\ &\quad= rem_{rt_1,\ldots,rt_k}(add_{rt,rt_1,1}(\cdots(add_{rt,rt_k,k}[ren_{rt\to rt,f}(\mathbf{rt})\oplus ren_{rt\to rt_1}(G_1)\oplus\cdots\\ &\qquad\cdots\oplus ren_{rt\to rt_k}(G_k)])\cdots))\end{aligned}$$

Intuitively, the graph denoted by $\mathscr{I}(f)(G_1,\ldots,G_k)$ is obtained as follows: one takes the union of disjoint copies of $G_1,\ldots,G_k$, and a new vertex say s that will be the root; one links by an i-edge s to the root of G_i, for each $i=1,\ldots,k$. The labels $rt_1,\ldots,rt_k$

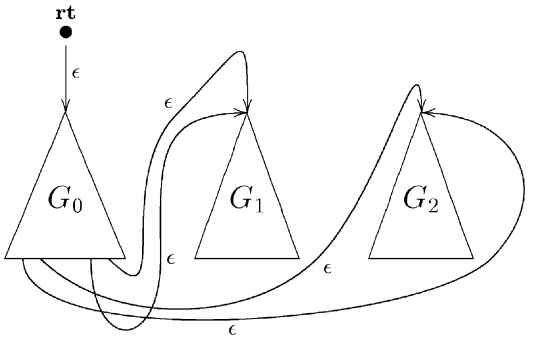


Fig. 2.

are "temporary". They are used for insuring correct connection of s with the roots of $G_1,\ldots,G_k$ by 1-,…,k-edges.

Before going on we can make an observation:

Claim 3.6. *For* $t\in T^{\infty}(F-\{\Omega\},X)$ *we have* $\mathscr{I}nt(t)=G(t)$.

It remains to define $\mathscr{I}(sub_{\bar{x}})$. We illustrate first the mapping $\mathscr{I}(sub_{x,y})$. With rooted graphs G_0,G_1,G_2, it associates the graph shown in Fig. 2. The ε-edges towards G_1 originate from the vertices of G_0 labelled by x (the first variable of $\bar{x}$), and those to G_2 from the vertices of G_0 labelled by y. The labels x,y in G_0 are removed. However G_0,G_1,G_2 may contain vertex labels in $X-\{x,y\}$ and G_1,G_2 may contain vertices labelled by x or y.

We now give the general definition of $\mathscr{I}(sub_{\bar{x}})$ where $\bar{x}=(x_1,\ldots,x_n)$. We use auxiliary labels $rt_0,\ldots,rt_n$ and $aux_1,\ldots,aux_n$. We define

$$\begin{aligned}&\mathscr{I}(sub_{\bar{x}})(G_0,G_1,\ldots,G_n)\\&\quad=rem_{aux_1,\ldots,aux_n,\ldots,rt_0,\ldots,rt_n}(add_{rt,rt_0,\varepsilon}(add_{aux_1,rt_1,\varepsilon}(\cdots(add_{aux_n,rt_n,\varepsilon}[\\&\qquad\mathbf{rt}\oplus ren_{rt\to rt_0}(ren_{x_1\to aux_1}(\cdots ren_{x_n\to aux_n}(G_0))\cdots))\\&\qquad\oplus ren_{rt\to rt_1}(G_1)\oplus\cdots\oplus ren_{rt\to rt_n}(G_n)])))\cdots))).\end{aligned}$$

We recall that we apply the operator Un to directed graphs having a unique vertex labelled by rt, (still called the root) but such that not necessarily all vertices are reachable from the root by a directed path.

Lemma 3.7. *For every* $t\in T^{\infty}(F\cup Sub(X),X)$ *we have*

$$G(Eval(t))=Contr_{\varepsilon}(Un(\mathscr{I}nt(t))).$$

Before proving this lemma, we show that $\mathscr{I}nt(t)$ is well-defined (which does not follow from Section 2.6) because we let $\mathscr{I}(\Omega)$ be **rt** and not $\emptyset$.

Fact 3.8. *For every* $t \in T(F \cup Sub(X), X)$ *we have* $\mathscr{I}nt(t) \in \mathscr{D}'_1(A, \{\varepsilon\}; P)$.

Proof. By induction on the structure of t. This is true for $t = \Omega$ since we have $\mathscr{I}(\Omega) =$ **rt**. The other cases follow from the definition of $\mathscr{I}(w)$ for the other symbols in $F \cup Sub(X) \cup X$. □

Fact 3.9. *If* $t, t' \in T(F \cup Sub(X), X)$ *and* $t \prec t'$ *then* $\mathscr{I}nt(t) \subseteq \mathscr{I}nt(t')$.

Proof. By induction on the structure of t using the fact that $\mathbf{rt} = \mathscr{I}(\Omega) \subseteq G$ for every $G \in \mathscr{D}_1(A \cup \{\varepsilon\}; P)$. □

Hence we can define $\mathscr{I}nt(t)$ for $t \in T^{\infty}(F \cup Sub(X), X)$ as the union of the increasing sequences of graphs $\mathscr{I}nt(t_n)$ where $t_0 \prec t_1 \prec, \ldots, \prec t_n \prec \ldots$, each t_n is finite and $t = \sup_n(t_n)$.

Proof of Lemma 3.7. *First part*: *By induction on t for t finite*. The cases where $t \in \{\Omega\} \cup X \cup F_0$ are clear. Two cases remain.

Case 1: $t = f(t_1, \ldots, t_k)$.

We have $Eval(t) = f(Eval(t_1), \ldots, Eval(t_k))$ hence, by the definition of $\mathscr{I}(f)$ we have

$$G(Eval(t)) = \mathscr{I}(f)(G(Eval(t_1)), \ldots, G(Eval(t_k))). \tag{3.1}$$

On the other hand:

$$Un(\mathscr{I}nt(t)) = Un(\mathscr{I}(f)(\mathscr{I}nt(t_1), \ldots, \mathscr{I}nt(t_k))) = \mathscr{I}(f)(\mathscr{I}nt(t_1), \ldots, \mathscr{I}nt(t_k))$$

by the way $\mathscr{I}(f)$ is defined (intuitively, it is a tree-construction operation, hence is "invariant by unfolding"); similarly, since $\mathscr{I}(f)$ introduces no ε-edge, it is invariant by ε-edge contractions, hence:

$$\begin{aligned} Contr_{\varepsilon}(Un(\mathscr{I}nt(t))) &= Contr_{\varepsilon}(\mathscr{I}(f)(Un(\mathscr{I}nt(t_1)), \ldots, Un(\mathscr{I}nt(t_k)))) \\ &= \mathscr{I}(f)(Contr_{\varepsilon}(Un(\mathscr{I}nt(t_1))), \ldots, Contr_{\varepsilon}(Un(\mathscr{I}nt(t_k)))). \end{aligned}$$

Hence

$$Contr_{\varepsilon}(Un(\mathscr{I}nt(t))) = \mathscr{I}(f)(G(Eval(t_1)), \ldots, G(Eval(t_k)))$$

by using induction, hence is equal to $G(Eval(t))$ by (3.1).

Case 2: $t = sub_{\vec{x}}(t_0, t_1, \ldots, t_n)$.

From the definition of $\mathscr{I}(sub_{\vec{x}})$ it is clear that

$$G(Eval(t)) = Contr_{\varepsilon}(Un(\mathscr{I}(sub_{\vec{x}})(G(Eval(t_0)), \ldots, G(Eval(t_k))))).$$

Hence, using induction, we need only prove that for $H_0,\dots,H_k \in \mathscr{D}'_1(A,\{\varepsilon\};P)$ we have

$$\begin{aligned} &Contr_\varepsilon(Un(\mathscr{I}(sub_{\bar{x}})(H_0,H_1,\dots,H_k))) \\ &\quad = Contr_\varepsilon(Un(\mathscr{I}(sub_{\bar{x}})(Contr_\varepsilon(Un(H_0)),\dots,Contr_\varepsilon(Un(H_k))))). \end{aligned} \tag{3.2}$$

We have actually:

$$Un(\mathscr{I}(sub_{\bar{x}})(H_0,\dots,H_k)) = Un(\mathscr{I}(sub_{\bar{x}})(Un(H_0),\dots,Un(H_k))). \tag{3.3}$$

(clear from the definition of $\mathscr{I}(sub_{\bar{x}})$ and the construction of Un used in Proposition 2.9). No more difficult is to prove that

$$\begin{aligned} &Contr_\varepsilon(Un(\mathscr{I}(sub_{\bar{x}})(Un(H_0),\dots,Un(H_k)))) \\ &\quad = Contr_\varepsilon(Un(\mathscr{I}(sub_{\bar{x}})(Contr_\varepsilon(Un(H_0)),\dots,Contr_\varepsilon(Un(H_k))))). \end{aligned} \tag{3.4}$$

Hence (3.3) and (3.4) yield (3.2), which completes the proof of the first part.

Second part: *Extension to t infinite*. We let $t=\sup_n(t_n)$ where t_n is an increasing sequence of finite terms, $t_n \in T(F\cup Sub(X),X)$. By the definition of $Eval$ we have $Eval(t)=\sup_n(Eval(t_n))$. The graphs $G(Eval(t_n))$ form an increasing sequence for inclusion and $G(Eval(t))$ is the union of these graphs.

On the other side, we have to check that $Contr_\varepsilon(Un(\mathscr{I}nt(t)))$ is the least upper bound of the graphs $Contr_\varepsilon(Un(\mathscr{I}nt(t_n)))$ which also form an increasing sequence since they are equal respectively to $G(Eval(t_n))$ for each n.

Since for $G\in\mathscr{D}'_1(A;P)$, $Un(G)$ has for vertices the finite paths in G with origin $root_G$, it is clear that if $G\subseteq G'$, then $Un(G)\subseteq Un(G')$. Furthermore, if $G_0\subseteq G_1\subseteq\cdots\subseteq G_n\subseteq\cdots$ then $Un(\bigcup_{n\geqslant 0}G_n)=\bigcup_{n\geqslant 0}Un(G_n)$. Hence $Un(\mathscr{I}nt(t))$ is the union of the increasing sequence of graphs $Un(\mathscr{I}nt(t_n))$, $n\geqslant 0$.

It is not true in general that $Contr_\varepsilon(G)\subseteq Contr_\varepsilon(G')$ if $G\subseteq G'$ (because if G' is G augmented with more ε-edges, $Contr_\varepsilon(G')$ may have less vertices than $Contr_\varepsilon(G)$). However this is true for G,G' if they are trees, as one checks easily. Furthermore, $Contr_\varepsilon$ is also ω-continuous on trees. Hence the least upper bound (for inclusion) of the increasing sequence $Contr_\varepsilon(Un(\mathscr{I}nt(t_n)))$ is $Contr_\varepsilon(Un(\mathscr{I}nt(t)))$ as was to be proved. □

4. Regular, algebraic and hyperalgebraic trees

We review the definitions of regular, algebraic and hyperalgebraic trees, and we apply the main result of Section 3 to these trees and to the recursive definitions they represent.

4.1. Regular trees

We review results from [3,4]. A *regular* system of equations over $T^\infty(F,X)$ is an n-tuple of the form:

$$\langle t_i = sub_{\bar{x}}(s_i, t_1, \ldots, t_n); 1 \leqslant i \leqslant n\rangle, \tag{4.1}$$

where $\bar{x}=(x_1,\ldots,x_n), s_i \in T(F,X)$ and each t_i is an unknown. It has a least solution in $T^\infty(F,X)$. (We assume that $\Omega \in F$ so that $T^\infty(F,X)$ is partially ordered by $\prec$ and ω-complete.) Note that here $sub_{\bar{x}}$ is evaluated and handled as a mapping on $T^\infty(F,X)$. A regular system of the form (4.1) above has also a least (and actually unique) solution in $T^\infty(F \cup Sub(X),X)$ where the operations $sub_{\bar{x}}$ are *unevaluated* and treated as those in F. We let $(t'_1,\ldots,t'_n)$ be this solution. By a fundamental result of Mezei and Wright [4, Lemma 5.3], least solutions of regular systems are preserved under ω-continuous mappings. Applying this to *Eval* we obtain that $Eval(t'_i)=t_i$ for each $i=1,\ldots,n$.

A term $t \in T^\infty(F,X)$ is *regular* iff it is a component of the solution of such a system. We denote by $REG(F,X)$ the set of these terms. We will call them *regular trees* in order to keep the well-known terminology. Other characterizations of regular trees are given in [3].

By routine transformations of regular systems, one can define regular trees in $REG(F,X)$ by regular systems in *normal form* i.e., of the form $\langle t_i = e_i; 1 \leqslant i \leqslant n\rangle$, where for each i,

- either $e_i \in X \cup F_0$
- or $e_i = f(t_{i_1},\ldots,t_{i_k})$ for some k, some $f \in F_k$, some $i_1,\ldots,i_k \in \{1,\ldots,n\}$.

Example 4.1. The system

$$t_1 = sub_{u,v}(f(x, f(u,v)), t_1, t_2),$$

$$t_2 = sub_{u,v}(g(z, f(u,v)), t_1, t_2),$$

can be written more concretely as

$$t_1 = f(x, f(t_1, t_2)),$$

$$t_2 = g(z, f(t_1, t_2))$$

and, can be replaced by the following system in normal form:

$$t_1 = f(t_3, t_4),$$

$$t_2 = g(t_5, t_4),$$

$$t_3 = x,$$

$$t_4 = f(t_1, t_2),$$

$$t_5 = z.$$

The monadic second-order theory of a regular tree t is decidable (by Rabin's theorem, see [21]). This means that one can decide whether a given MS-formula φ holds in t.

Since we represent a term t by the structure $G(t)$, we write $G(t) \models \varphi$ although $G(t)$ is not the standard structure for representing t. However, $G(t)$ and the classical structure with k_F successors are definable in each other by MS-formulas. Hence, our definition of "t has a decidable MS theory" is equivalent to the classical one.

Some nonregular infinite trees also have decidable MS theories: see the introduction for references. Our objective is precisely to define such trees.

4.2. Second-order substitution

We recall from [3] the notion of second-order substitution.

We let Φ be a set of function variables, each of them given with a fixed arity ($\rho(\varphi) \in \mathbf{N}$ for $\varphi \in \Phi$). Our intention is to define the result of the substitution in $t \in T^\infty(F \cup \Phi, X)$ of s for φ where the variables $x_1, \ldots, x_m$ of s correspond to the 1st, …, mth argument of φ. In order to specify this sequence we will write that we substitute $\lambda x_1, \ldots, x_m . s$ for φ or that we substitute s for $\langle \varphi, x_1, \ldots, x_m \rangle$. We let σ be a sequence $\langle \langle \varphi_1, w_1 \rangle, \ldots, \langle \varphi_n, w_n \rangle \rangle$ where $\varphi_1, \ldots, \varphi_n \in \Phi$ and are pairwise distinct and for each i, w_i is a sequence of pairwise distinct variables in X of length $\rho(\varphi_i)$. From σ we define a second-order substitution operation:

$$SUB_\sigma = T^\infty(F \cup \Phi, X)^{n+1} \to T^\infty(F \cup \Phi, X)$$

as follows. To simplify the notation we assume that $\Phi = \{\varphi_1, \ldots, \varphi_n\}$. We first define $SUB_\sigma(t, t_1, \ldots, t_n)$ for $t \in T(F \cup \Phi, X), t_i \in T^\infty(F \cup \Phi, X)$ by induction on the structure of t:

$$SUB_\sigma(a, t_1, \ldots, t_n) = a \quad \text{if } a \in \{\Omega\} \cup X \cup F_0,$$

$$SUB_\sigma(f(s_1, \ldots, s_k), t_1, \ldots, t_n) = f(SUB_\sigma(s_1, t_1, \ldots, t_n), \ldots, SUB_\sigma(s_k, t_1, \ldots, t_n))$$

$$\text{if } f \in F_k, \ k \geqslant 1,$$

$$SUB_\sigma(\varphi_i(s_1, \ldots, s_m), t_1, \ldots, t_n)$$

$$= sub_{w_i}(t_i, SUB_\sigma(s_1, t_1, \ldots, t_n), \ldots, SUB_\sigma(s_m, t_1, \ldots, t_n)),$$

where $\langle \varphi_i, w_i \rangle$ is the ith pair in σ for $1 \leqslant i \leqslant n$.

Example 4.2. We let

$$\sigma = \langle \langle \varphi, x, z, u \rangle, \langle \theta, y, z, u \rangle \rangle,$$

$$t = f(\varphi(a, \theta(a, x, g(x, z)), b)),$$

$$t_1 = h(x, z), \text{ and}$$

$$t_2 = k(m(x, y), p(z, u)).$$

Then $SUB_\sigma(t,t_1,t_2)$ is the term

$$\begin{aligned} t' &= f(sub_{x,z,u}(t_1, SUB_\sigma(a,t_1,t_2), SUB_\sigma(\theta(a,x,g(x,z)),t_1,t_2), SUB_\sigma(b,t_1,t_2))) \\ &= f(h(SUB_\sigma(a,t_1,t_2), SUB_\sigma(\theta(a,x,g(x,z)),t_1,t_2))) \\ &= f(h(a, sub_{y,z,u}(t_2, SUB_\sigma(a,t_1,t_2), SUB_\sigma(x,t_1,t_2), SUB_\sigma(g(x,z),t_1,t_2))) \\ &= f(h(a, sub_{y,z,u}(t_2,a,x,g(x,z)))) \\ &= f(h(a,k(m(x,a),\, p(x,g(x,z))))). \end{aligned}$$

We make a few observations. The term t_1 that we substitute for φ has no occurrence of u. Hence the third argument of φ in t, namely b disappears. The variable x does not appear in the argument list of θ (in σ). Hence it is treated in t_2 as a constant, and we find it as first argument of m, as it is in t_2. The other occurrences of x, as arguments of p and of g "come from" the second and the third argument of θ in t.

Lemma 4.3. *For each σ the mapping SUB_σ is monotone in its first argument, and ω-continuous in the other ones.*

Proof. Standard proof by induction on the structure of the first argument. □

It follows that SUB_σ extends into a continuous mapping: $T^\infty(F\cup\Phi,X)^{n+1}\to T^\infty(F\cup\Phi,X)$ in a standard way.

Remark 4.4. Let φ be unary, let φ^ω be the term $\varphi(\varphi(\varphi(\cdots)))\in T^\infty(\{\varphi\})$, let $\sigma=\langle\langle\varphi,x\rangle\rangle$. Then $SUB_\sigma(\varphi^\omega,x)=\Omega$. It is easy to check that $SUB_\sigma(\varphi^n(\Omega),x)=\Omega$ for every n whence the observation. Hence, even if Ω does not appear in any argument of SUB_σ, it may appear in the result.

4.3. Algebraic trees

For each $m\geqslant 1$, we denote by X_m the "standard" set of variables $X_m=\{x_1,\ldots,x_m\}$. An *algebraic system of equations* on $T^\infty(F,X_m)$ is a system, written with a set $\Phi=\{\varphi_1,\ldots,\varphi_n\}$ of function symbols with $\rho(\varphi_i)\leqslant m$, of the form:

$$S=\langle\varphi_i(x_1,\ldots,x_{\rho(\varphi_i)})=s_i; 1\leqslant i\leqslant n\rangle,$$

where for each i, $s_i\in T(F\cup\Phi,X_{\rho(\varphi_i)})$. A *solution* of S is an n-tuple of terms $(t_1,\ldots,t_n)$, where $t_i\in T^\infty(F,X_{\rho(\varphi_i)})$ such that for each i:

$$t_i=SUB_\sigma(s_i,t_1,\ldots,t_n),$$

where $\sigma=\langle\langle\varphi_1,w_1\rangle,\ldots,\langle\varphi_n,w_n\rangle\rangle$ and for each $j=1,\ldots,n$ we have $w_j=(x_1,\ldots,x_{\rho(\varphi_j)})$. Such a system has a least solution in $T^\infty(F,X_n)^n$. An *algebraic tree* is a component of the least solution of an algebraic system. We denote the corresponding set by $ALG(F,X_m)$; (it is a subset of $T^\infty(F,X_m)$).

Algebraic systems have been introduced as syntactic base of *recursive applicative program schemes* (see [5,11]).

Proposition 4.5. *Every algebraic tree* $t \in ALG(F,X)$ *is* $Eval(t')$ *for some regular tree* $t' \in REG(F \cup Sub(X'),X')$, *with* X' *finite* $X' \supseteq X$. *A regular system defining* t' *can be constructed from an algebraic system defining* t.

Proof. Let $t \in ALG(F,X)$ be given as the first component of the least solution of an algebraic system. It is also the first component of the least solution of an algebraic system of the special "normal" form (that one can construct from the given one)

$$S = \langle \varphi_i(x_1,\ldots,x_{\rho(\varphi_i)}) = s_i; 1 \leqslant i \leqslant n \rangle$$

such that each s_i is:
(i) either x_j $(1 \leqslant j \leqslant \rho(\varphi_i))$,
(ii) or $f(\varphi_{i_1}(x_1,\ldots,x_{\rho(\varphi_{i_1})}),\ldots,\varphi_{i_k}(x_1,\ldots,x_{\rho(\varphi_{i_k})}))$,
(iii) or $\varphi_j(\varphi_{i_1}(x_1,\ldots,x_{\rho(\varphi_{i_1})}),\ldots,\varphi_{i_k}(x_1,\ldots,x_{\rho(\varphi_{i_k})}))$ with $1 \leqslant j,i_1,\ldots,i_k \leqslant n, k = \rho(\varphi_j)$.
A solution of S is thus an n-tuple $(t_1,\ldots,t_n)$ of terms in $T^\infty(F,X_m)$ (with $Var(t_i) \subseteq X_{\rho(\varphi_i)}$) which satisfies the equations:

$$S' = \begin{cases} t_i = x_j & \text{(if we are in case (i))}, \\ t_i = f(t_{i_1},\ldots,t_{i_k}) & \text{(if we are in case (ii))}, \\ t_i = sub_{x_1,\ldots,x_k}(t_j,t_{i_1},\ldots,t_{i_k}) & \text{(if we are in case (iii))}. \end{cases}$$

Note that the terms $t_1,\ldots,t_n$ defined by S' belong to $T^\infty(F,X_m)$, not to $T^\infty(F \cup Sub(X_m),X_m)$, hence that the operations $sub_{x_1,\ldots,x_k}$ are evaluated.

Let us denote by $(t'_1,\ldots,t'_n)$ the unique solution of S' in $T^\infty(F \cup Sub(X_m),X_m)$ (without "evaluating" $sub_{x_1,\ldots,x_k}$, see Section 4.1). Hence each t'_i belongs to $REG(F \cup Sub(X_m),X_m)$ and its image in $T^\infty(F,X_m)$ under $Eval$ is t_i according to the Mezei–Wright Theorem. □

As a corollary we obtain another proof of Theorem 5.2 of [8]. □

Theorem 4.6. *The monadic second-order theory of an algebraic tree is decidable.*

Proof. Let $t_1 \in ALG(F,X)$ be defined by an algebraic system. We have $t_1 = Eval(t'_1)$ where $t'_1 \in REG(F \cup Sub(X),X)$ is defined by a regular system that one can construct from the one defining t_1. The MS theory of t'_1 is thus decidable. Since $Eval$ is MS-compatible by Theorem 3.3, the MS theory of t_1 reduces to that of t'_1 hence is decidable. □

Example 4.7. Consider the equation

$$\varphi_1(x_1,x_2) = f(x_1,\varphi_1(g(x_2),\varphi_1(x_1,x_2))).$$

We first transform it into a system of the appropriate "normal form":

$$\varphi_1(x_1,x_2) = f(\varphi_2(x_1),\varphi_3(x_1,x_2)),$$
$$\varphi_2(x_1) = x_1,$$
$$\varphi_3(x_1,x_2) = \varphi_1(\varphi_4(x_1,x_2),\varphi_1(x_1,x_2)),$$
$$\varphi_4(x_1,x_2) = g(\varphi_5(x_1,x_2)),$$
$$\varphi_5(x_1,x_2) = x_2.$$

The corresponding regular system over $T^\infty(F \cup Sub(X_2),X_2)$ is thus:

$$t'_1 = f(t'_2,t'_3),$$
$$t'_2 = x_1,$$
$$t'_3 = sub_{x_1,x_2}(t'_1,t'_4,t'_1),$$
$$t'_4 = g(t'_5),$$
$$t'_5 = x_2.$$

Of course the regular tree $t'_1 \in REG(F \cup Sub(X_2),X_2)$ could be defined directly by:

$$t'_1 = f(x_1, sub_{x_1,x_2}(t'_1, g(x_2), t'_1)).$$

The notion of a system in normal form has been used to simplify the formal construction. Here is another example.

Example 4.8. Consider the following equation:

$$\varphi_1(x_1) = \varphi_1(f(x_1)).$$

Any term $t \in T(F,X)$ without an occurrence of x_1 is a solution. The least solution is Ω. The above equation can be translated into the regular equation:

$$t'_1 = sub_{x_1}(t'_1, f(x_1)).$$

The regular tree t'_1 corresponding to its unique solution in $T^\infty(F \cup \{sub_{x_1}\},X)$ has a left-most infinite branch with only the symbol sub_{x_1}. Hence t'_1 evaluates by *Eval* into Ω.

We now state the following proposition which is the converse of Proposition 4.5.

Proposition 4.9. *For every finite set X of variables, we have*

$$Eval(REG(F \cup Sub(X),X)) \subseteq ALG(F,X).$$

Proof. Let $r \in REG(F \cup Sub(X),X)$, where $X = \{x_1,\ldots,x_n\}$. As a regular tree, r is a component of the unique solution in $T^\infty(F \cup Sub(X),X)$ of a finite system of equations

$$\langle t_i = e_i; 1 \leqslant i \leqslant m \rangle, \tag{4.2}$$

where for each i,
- either $e_i \in X$,
- or $e_i \in F_0$,
- or $e_i = f(t_{i_1}, \ldots, t_{i_k})$ for some $k \geqslant 1$, some $f \in F_k$, some $i_1, \ldots, i_k \in \{1, \ldots, n\}$,
- or $e_i = sub_{\bar{x}}(t_j, t_{i_1}, \ldots, t_{i_k})$ for some sequence $\bar{x}$ of pairwise distinct elements of X, some $j, i_1, \ldots, i_k \in \{1, \ldots, n\}$.

Without loss of generality we can assume that $\bar{x}$ is an increasing subsequence of $(x_1, \ldots, x_n)$, because if necessary, we can permute the sequence $(i_1, \ldots, i_k)$ in order to replace $\bar{x}$ by an increasing sequence. Moreover, we can also assume that $\bar{x} = (x_1, \ldots, x_n)$. Indeed, if a variable x_j is missing, we add it and we let x_j to be the term that should be substituted to it. We illustrate this construction on the following example.

Let $n = 5$. An equation of the form $t = sub_{x_5, x_1, x_3}(t_0, t_1, t_2, t_3)$ is replaced by the equation $t = sub_{x_1, x_2, x_3, x_4, x_5}(t_0, t_2, x_2, t_3, x_4, t_1)$.

Once we agreed on this transformation, the last case of the definition a system of equations is $e_i = sub_{\bar{x}}(t_j, t_{i_1}, \ldots, t_{i_n})$ and $\bar{x} = (x_1, \ldots, x_n)$.

For each $i = 1, \ldots, m$, we introduce a function variable φ_i of arity n and we define the algebraic system

$$\langle \varphi_i(x_1, \ldots, x_n) = s'_i; 1 \leqslant i \leqslant m \rangle, \tag{4.3}$$

where in the above four cases we have now, respectively,
- $s'_i = e_i$ if $e_i \in X$,
- $s'_i = e_i$ if $e_i \in F_0$,
- $s'_i = f(\varphi_{i_1}(\bar{x}), \ldots, \varphi_{i_k}(\bar{x}))$ if $e_i = f(t_{i_1}, \ldots, t_{i_k})$,
- $s'_i = \varphi_j(\varphi_{i_1}(\bar{x}), \ldots, \varphi_{i_k}(\bar{x}))$ if $e_i = sub_{\bar{x}}(t_j, t_{i_1}, \ldots, t_{i_k})$,

where $\bar{x}$ is fixed as $(x_1, \ldots, x_n)$ by the above agreement.

Let $(r_1, \ldots, r_m)$ (resp. $(r'_1, \ldots, r'_m)$) be the unique solution of (4.2) (resp. (4.3)) in $T^\infty(F \cup Sub(X), X)$ (resp. $T^\infty(F, X)$). It is clear that $Eval(r_i) = r'_i$. It follows in particular that t is an algebraic tree. □

4.4. Damm's hierarchy

We have seen that $Eval(REG) = ALG$ (leaving functions and variables unspecified). Provided suitable alphabets are used, we can define the family of level-n hyperalgebraic (called n-rational in [15]) trees roughly as

$$ALG^n = Eval^n(REG).$$

More precisely, we have already, for a finite set F of function symbols and a finite set X of variables:

$$ALG(F, X) = \left(\bigcup_{\substack{Y \supseteq X \\ Y \text{finite}}} Eval(REG(F \cup Sub(Y), Y)) \right) \cap T^\infty(F, X).$$

(In function definitions we must allow for sets Y of "parameters" that are larger than X but still finite.) We define

$$ALG^0(F,X) = REG(F,X),$$

$$ALG^n(F,X) = T^\infty(F,X) \cap \left(\bigcup_{\substack{Y \supseteq X \\ \text{finite}}} Eval(ALG^{n-1}(F \cup Sub(Y), Y)) \right),$$

so that $ALG^1(F,X) = ALG(F,X)$.

Theorem 4.10. *Every tree in ALG^n has a decidable MS theory.*

Proof. This is straightforward since *Eval* is MS-compatible. One uses an induction on n. □

These trees have been considered by Damm in [15] where it is established that ALG^n forms a strict hierarchy.

The next question is to understand which types of recursive definitions these trees represent. To this end, we consider explicit composition of functions of arbitrary arity. This is an extension of the usual composition of unary functions "$\circ$". For each $n \geqslant 1$ we denote by $comp_n$ the overloaded functional operator such that $h = comp_n(f, g_1, \ldots, g_n)$ is well-defined iff

$$f : B^n \to A \quad \text{and}$$

$$g_i : C \to B \quad \text{for each } i = 1, \ldots, n$$

are total functions and A, B, C are sets. The term $comp_n(f, g_1, \ldots, g_n)$ denotes the function $\lambda x \cdot f(g_1(x), \ldots, g_n(x))$. It should be clear that if $t \in T^\infty(F, \{x_1, \ldots, x_n\})$, $s_i \in T^\infty(F, \{x_1, \ldots, x_k\})$ and A, B, D are appropriate domains such that $val(t) : B^n \to A$ and $val(s_i) : D^k \to B$, then

$$val(sub_{(x_1,\ldots,x_n)}(t, s_1, \ldots, s_n)) = comp_n(val(t), val(s_1), \ldots, val(s_n)).$$

Hence $comp_n$ is the semantical meaning of the substitution operation $sub_{\bar{x}}$ if $\bar{x}$ has length n.

Having this in mind, we can consider recursive definitions as follows. We let $\mathbf{b}$ denote a "base" type (typically integers). We consider symbols of the following types:

$$c : \mathbf{b} \qquad v, u, x, y : \mathbf{b}$$

$$f : \mathbf{b} \times \mathbf{b} \to \mathbf{b} \qquad \varphi, \psi : \mathbf{b} \to \mathbf{b}$$

$$g, h : \mathbf{b} \to \mathbf{b} \qquad H : (\mathbf{b} \to \mathbf{b}) \times (\mathbf{b} \to \mathbf{b}) \times \mathbf{b} \times \mathbf{b} \to \mathbf{b}$$

We let H be defined by the following recursive definition

$$H(\varphi,\psi,x,y) = f(x,\varphi(H(\lambda u.\varphi(\psi(u)),\lambda v.f(\varphi(v),\psi(h(v))),g(y),\varphi(c)))). \tag{4.4}$$

It involves parameters of two types: $\mathbf{b}$ and $\mathbf{b}\to\mathbf{b}$. Hence it is not an algebraic system.

Let us introduce a new symbol $H':(\mathbf{b}\to\mathbf{b})\times(\mathbf{b}\to\mathbf{b})\to(\mathbf{b}\times\mathbf{b}\to\mathbf{b})$ defining a functional such that

$$H'(\varphi,\psi) = comp_2(f,\pi_1,comp_1(\varphi,comp_2(H'(comp_1(\varphi,\psi),$$
$$comp_2(f,\varphi,comp_1(\varphi,h))),comp_1(g,\pi_2),comp_1(\varphi,c_2)))) \tag{4.5}$$

where

$\pi_1:\mathbf{b}\times\mathbf{b}\to\mathbf{b}$ is the first projection,
$\pi_2:\mathbf{b}\times\mathbf{b}\to\mathbf{b}$ is the second projection,
$c_2:\mathbf{b}\times\mathbf{b}\to\mathbf{b}$ is the constant function $\lambda xy\cdot c$.

The definition of H' involves the following types:

$$\mathbf{b}' = \mathbf{b}\to\mathbf{b}$$

$$\mathbf{b}'' = \mathbf{b}\times\mathbf{b}\to\mathbf{b}$$

so that

$$H' : \mathbf{b}'\times\mathbf{b}'\to\mathbf{b}'' \qquad f : \mathbf{b}''$$
$$comp_1 : \mathbf{b}'\times\mathbf{b}'\to\mathbf{b}' \qquad g,h : \mathbf{b}'$$
$$comp_2 : \mathbf{b}''\times\mathbf{b}'\times\mathbf{b}'\to\mathbf{b}' \qquad \pi_1,\pi_2,c_2 : \mathbf{b}''$$

and clearly H' is defined by an algebraic equation over basic functions f, g, h, $comp_1$, $comp_2$, π_1, π_2, c_2.

The tree $T(H)\in T^\infty(\{f,g,h,\varphi,\psi\},\{x,y,c\})$ is thus $Eval(T(H'))$ where in addition

- $Eval(\pi_1)=x$ and $Eval(\pi_2)=y$,
- $Eval$ treats $comp_2$ as $sub_{\bar{x}}$ with $\bar{x}=(x,y)$ and $comp_1$ similarly with $\bar{x}=(x)$. [1]

Hence $T(H)\in ALG^2(\{f,g,h,\varphi,\psi\},\{x,y,c\})$.

It is not yet completely clear if any recursive definition like (4.4) can be translated into an algebraic equation like (4.5). The problem comes with λ-schemes. Here is an example from [19] (called "unsafe" there).

$$H(\varphi,x) = f(x,\varphi(H(\lambda u\cdot f(\varphi(u),\underline{x}),x))). \tag{4.6}$$

The underlined occurrence of x is free in the λ-subterm and this raises a problem with the translation. In the present example one can observe that (4.6) is equivalent to

$$H(\varphi,x) = H_1(\lambda uv\cdot\varphi(u),x),$$
$$H_1(\psi,x) = f(x,\psi(H_1(\lambda uv\cdot f(\psi(u,v),v),x),x)), \tag{4.7}$$

[1] We write the definition with *comp* in order to stress semantics.

because H_1 (with $\psi : \mathbf{b} \times \mathbf{b} \to \mathbf{b}$) is such that

$$H_1(\psi, x) = H(\lambda u \cdot \psi(u, x), x).$$

(This latter fact can be proved from (4.6) and (4.7) by the standard "Scott's induction method".) It follows that the tree associated with H defined by (4.6) is in ALG.[1] However, we leave as an open question whether a similar transformation can be applied for all "λ-schemes" with free variables in λ-terms. This question concerns level-2 schemes. What about higher levels?

5. Monadic second-order logic and program schemes

Our initial motivation was to decide properties of program schemes expressible in MS logic.

Consider a program scheme the semantics of which is represented by an infinite term T in $T^{\infty}(F, X)$. It may be the case that a variable x does not occur in T. This corresponds to the fact that the function defined by the scheme does not actually depend on the argument corresponding to x. A simplification of the writing of the scheme may be obtained from this observation. Such transformations are considered in [17]. Whether or not x occurs in T is not necessarily easy to decide if the program scheme consists of several mutually recursive definitions using parameters of function or functional type. Hence, even if the existence of an occurrence of x in T is a first-order property over the relational structure representing T, its expression in terms of the syntax of the scheme is not necessarily easy. However, our result states that it is decidable for the schemes corresponding to level-n hyperalgebraic trees.

Here is an example of a related monadic second-order but nonfirst-order property. We consider the finiteness of the number of occurrences in T of a variable x (or of a function symbol). That this property is monadic second-order can be seen as follows. Let us say that a set of nodes W in an infinite tree is a *cut* if no two nodes of W are on a same branch and every maximal branch contains one (and thus only one) node of W. Since the trees corresponding to terms have nodes of finite (even bounded) degree, every cut in such a tree is finite (by Koenig's Lemma). Hence a set of nodes U is finite iff there is a cut W such that every node of U is between the root and an element of W. This latter condition is expressible in MS logic.

Hence, as an application of our result that is relevant to the static analysis of recursive definitions, we get that, for every level-n hyperalgebraic tree, the finiteness of the number of occurrences of a given symbol is decidable.

References

[1] A. Asperti, V. Danos, C. Laneve, L. Regnier, Paths in the lambda-calculus, in: Proc. 9th IEEE Symp. on Logic in Comput. Sci., 1994, pp. 426–436.

[2] A. Asperti, S. Guerrini, The optimal implementation of functional programming languages, in: Cambridge Tracts in Theoretical Computer Science, Vol. 45, Cambridge University Press, Cambridge, 1998.

[3] B. Courcelle, Fundamental properties of infinite trees, Theoretical Comput. Sci. 25 (1983) 95–169.
[4] B. Courcelle, Equivalence and transformations of regular systems—applications to recursive program schemes and grammars, Theoretical Comput. Sci. 42 (1) (1986) 1–122.
[5] B. Courcelle, Recursive applicative program schemes, in: J. van Leeuwen (Ed.), Formal Models and Semantics, Handbook of Theoretical Computer Science, Vol. B, Elsevier, 1990, pp. 459–492.
[6] B. Courcelle, The monadic second-order logic of graphs, VII: Graphs as relational structures Theoretical Comput. Sci. 101 (1992) 3–33.
[7] B. Courcelle, Monadic second-order definable graph transductions: a survey. Theoretical Comput. Sci. 126 (1994) 53–75.
[8] B. Courcelle, The monadic second-order theory of graphs IX: Machines and their behaviours. Theoretical Comput. Sci. 151 (1995) 125–162.
[9] B. Courcelle, The expression of graph properties and graph transformations in monadic second-order logic, in: G. Rozenberg (Ed.), Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1, World Scientific, 1997, pp. 313–400.
[10] B. Courcelle, J. Engelfriet, G. Rozenberg, Handle-rewriting hypergraph grammars, J. Comput. Syst. Sci. 46 (2) (1993) 218–270.
[11] B. Courcelle, M. Nivat, Algebraic families of interpretations, in: IEEE Annual Symposium on Foundations of Computer Science, Houston, 1976, pp. 137–146.
[12] B. Courcelle, M. Nivat, The algebraic semantics of recursive program schemes, in: J. Winkowski (Ed.), Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Vol. 64, Zakopane, 1978, pp. 16–30.
[13] B. Courcelle, S. Olariu, Upper bounds to clique width of graphs, Discrete Applied Mathematics 101 (1999) 77–114.
[14] 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.
[15] W. Damm, The IO- and OI-hierarchies, Theoretical Comput. Sci. 20 (2) (1982) 95–208.
[16] J. Engelfriet, Graph grammars and tree transducers, in: S. Tison (Ed), Trees in Algebra and Programming—CAAP'94, Edinburgh, Apr. 1994, Vol. 787, pp. 15–36.
[17] I. Guessarian, Program transformations and algebraic semantics, Theoretical Comput. Sci. 9 (1979) 39–65.
[18] I. Guessarian, Algebraic Semantics, Lecture Notes in Computer Science, Vol. 99, Springer, Berlin, 1991.
[19] T. Knapik, D. Niwiński, P. Urzyczyn, Deciding monadic theories of hyperalgebraic trees, in: S. Abramsky (Ed.), 5th Internat. Conf. on Typed Lambda Calculi and Applications, Lecture Notes in Computer Science, vol. 2044, Kraków, May 2001, pp. 253–267.
[20] M.O. Rabin, Decidability of second-order theories and automata on infinite trees, Transactions of the American Mathematical Society 141 (1969) 1–35.
[21] W. Thomas, Languages, automata and logic, in: G. Rozenberg, A. Salomaa (Eds.), Beyond Words, Handbook of Formal Languages, Vol. 3, Springer, Berlin, 1997, pp. 389–455.