Semantical evaluations as monadic second-order compatible structure transformations

Bruno Courcelle

Université Bordeaux 1, LaBRI

(based in part on joint work with T. Knapik,
Université de La Réunion, France)

Summary

1. Introduction : MS (Monadic Second-Order) logic and semantics
2. MS logic and MS transductions
3. MS compatible structure transformations
4. First-order substitution as a basic operation
5. Hyperalgebraic trees and recursive program schemes

1. Why MS (Monadic Second-Order) logic is interesting ?

It expresses significant graph properties,

graph optimization functions, and graph transformations.

Results : 1. Graph properties and functions expressed in MS logic are evaluable in linear time on classes of graphs having a certain hierarchical structure (bounded tree-width or clique-width).

2. Satisfiability of MS properties on infinite equational graphs or, on infinite sets of finite graphs described by context-free graph grammars is decidable.

3. MS transductions applied to sets of finite graphs preserves context-free-ness (special case: intersection with an MS definable set)

4. Strong analogy with rational languages, rational transductions and context-free languages : the context-free sets of graphs are the images of the set of finite binary trees under MS transductions.

5. MS logic subsumes several languages used for semantics of programs like μ-calculus or CTL .

Semantics

1. Semantics is a mapping from :

syntactical objects to values in semantical domains.

2. Syntactical objects are usually finite terms over finite signatures.

3. Semantical domains : sets of partial functions over sets, real numbers approximated by intervals,
complete partial orders, complete lattices, etc…

Here: syntactical and semantical objects will be considered as finite or infinite discrete logical structures.

Example : Recursive applicative program schemes

Syntax : finite systems of mutually recursive equations like

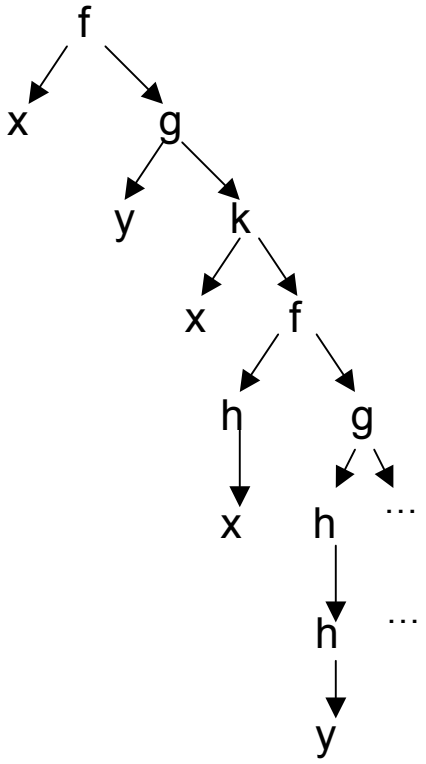
$\varphi(x,y) = f(x, g(y, \psi(x, h(y))))$

$\psi(x, y) = k(x, \varphi(h(x), h(y)))$

Semantical domain : Continous functions of appropriate types over complete partial orders.

By unfolding the recursion, one gets a pair of infinite trees

(one for φ, one for ψ) that represents faithfully all possible computations in all possible domains.



The tree for $\varphi(x,y)$.

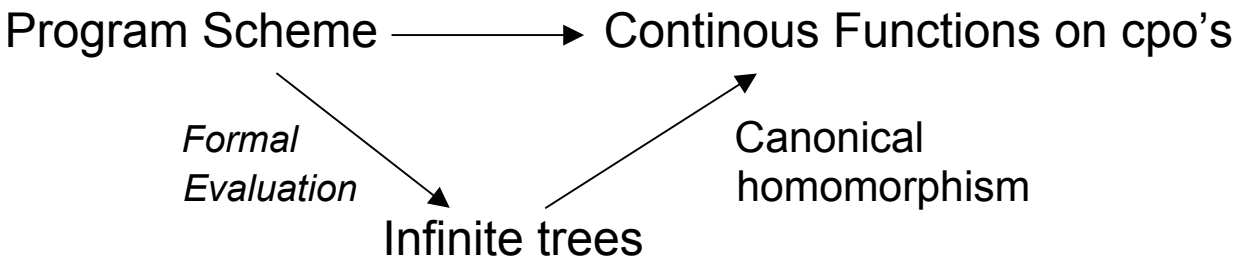
This pair of infinite trees can be considered as the semantical value.

The semantic mapping goes from :

(Finite) Systems of equations to tuples of infinite trees.

Both objects are representable by logical structures.

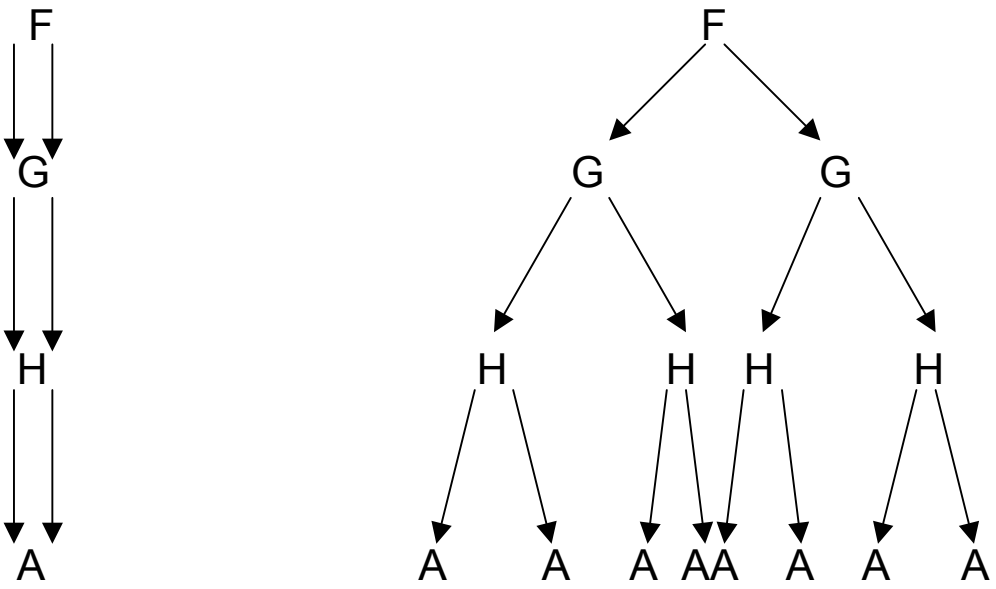
Semantical mapping :



The core of the semantical mapping is *the Formal Evaluation* step : unfolding and term substitutions

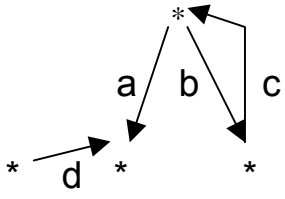
Semantics as a transformation of discrete (logical) structures.

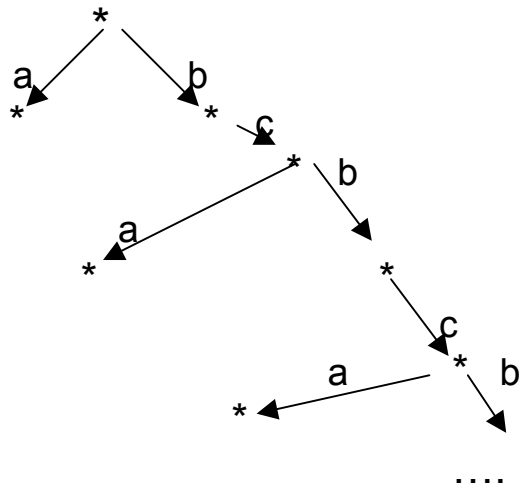
Examples : 1) Unsharing a directed acyclic graph into a term :



2) Unfolding a directed graph S (a transition system) into an infinite tree Unf(S).

Fact: Unf(S) is regular if S is finite.



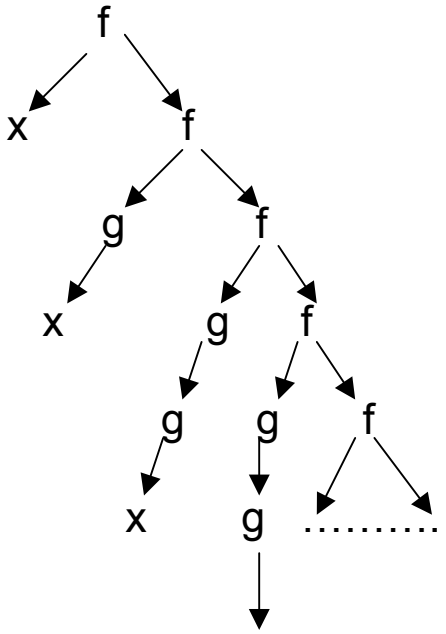


3) Unfolding a recursive applicative program scheme P

into a tuple of algebraic trees

Fact: Algebraic trees are not regular in general.

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



4) Hyperalgebraic trees : as in 3) with function variables

as parameters.

Thm (W. Damm) : There is a strict hierartchy :

Regular trees $\subset$ Algebraic trees $\subset$ level n -Hyperalgebraic

$H(\varphi, x, y) = f(x, \varphi(H(\varphi \circ \varphi , g(x), f(x, y)))$

$x, y : b$, $g, \varphi : b \to b$,

$f : b \times b \to b$, $H : (b \to b) \times b \times b \to b$

5) Evaluation of first-order substitution handled as a basic operation.

It is mapping from (finite) terms to (finite) terms.

$sub_{x,y}$ (s , t , t') evaluates to s [t / x, t' / y]

$sub_{x,y}$ is a new symbol, … [… / x, … / y] is an operation in metalanguage.

Example : Elimination of Substitution symbols

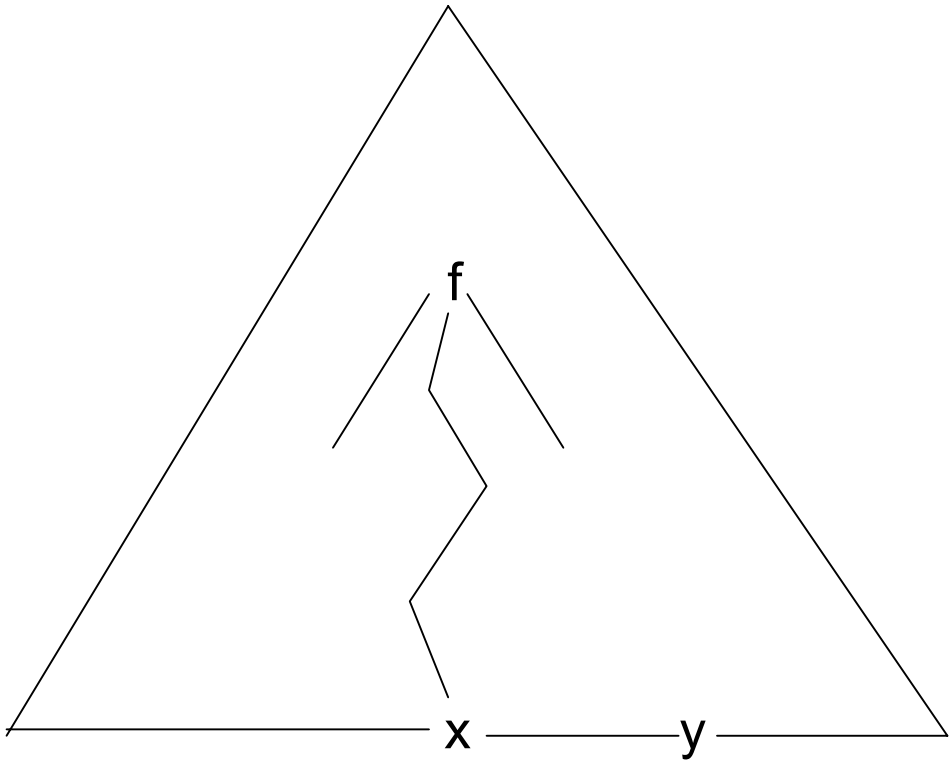
$sub_{x,y}$ (f(x, y, z), g(x,y), sub_z (f **(**z,z,x**)** , h(u)))

= $sub_{x,y}$ (f(x, y, z), g(x,y), f**(**h(u), h(u), x**)**)

= f(g(x,y), f**(**h(u), h(u),x**)**, z)

Question : Has MS logic anything to say about these semantical evaluations ?

Relevant monadic second-order properties of trees representing the behaviour of recursive program schemes



Does argument x (or y) occur anywhere below an occurrence of f ? (Tool for strictness analysis).

Has symbol f infinitely many occurrences in the tree ?

2. MS logic and MS transductions.

Logical expression of graph properties

Graphs are simple, directed, finite. (Extension is easy to undirected graphs, to hypergraphs, and to relational structures)

$G = < V, edg(.,.) >$ Vertices, edge relation

φ logical formula

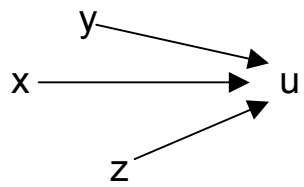
$G \models \varphi$ is a property of G

$G \models \varphi(x,y)$ is a property of a pair (x,y) of vertices of G

Example 1 : the first-order formula

$$\forall u,x,y,z \; (edg(x,u) \;\&\; edg(y,u) \;\&\; edg(z,u) \Rightarrow x = y \vee y = z \vee x = z)$$

expresses that G has indegree ≤ 2



Example 2: the monadic 2nd-order formula

$$\exists X,Y\ (\neg \exists x\ (x \in X \ \&\ x \in Y)$$
$$\&\ \forall x,y\ [edg(x,y) \Rightarrow \neg (x \in X \ \&\ y \in X)$$
$$\&\ \neg (x \in Y \ \&\ y \in Y)$$
$$\&\ (x \in X \vee x \in Y \vee y \in X \vee y \in Y)])$$

expresses 3-vertex colorability (color classes are X, Y, V(G) - (X $\cup$ Y))

Example 3 (important): the MS formula $\varphi(x,y)$

$$\exists X\ (x \in X \ \&\ \neg (y \in X)$$
$$\&\ \forall u,w\ (u \in X \ \&\ edg(u,w) \Rightarrow w \in X))$$

expresses that there is no directed path from x to y.

Example 4: The 2nd-order (non MS) formula

$$\exists R\ (\text{"R is a non-identity bijection on vertices"} \ \&$$
$$\forall u,w,u',w'[R(u,u') \ \&\ R(w,w') \Rightarrow$$
$$(edg\ (u,w) \Leftrightarrow edg(u',w')])$$

expresses the existence of a nontrivial auto-morphism.

Main examples of graph properties

First-order expressible:

degree at most (fixed) d

forbidden (fixed) subgraph

local properties (thm. by Gaifman):
boolean combinations of properties of the form "there are m disjoint 'balls' of radius r satisfying a certain first-order property"

Monadic Second-Order expressible properties

k - vertex colorability (fixed k ; NP-complete if $3 \leq k$)

transitive closures, "path" properties
like connectivity, cycles, trees

forbidden "minors" (Kuratowski Theorem)
whence planarity, genus at most fixed g

Properties of processes

Processes = Transition Systems

= Directed Labelled Graphs

Hierarchy of languages

On graphs:

CTL $\subset$ CTL* $\subset$ μ-Calculus $\subset$ MS-Logic

= Bisimulation-invariant MS properties

On binary trees:

CTL $\subset$ CTL* $\subset$ μ -Calculus = MS-Logic

MSL: $\forall X , \exists Y$ .....

M-Calculus: $x \in \mu Z \, \nu T$ .....

CTL* : On some infinite path some property holds infinitely often

MS-transductions

Definition of an MS-transduction (sometimes called an interpretation):

A transformation τ of structures, defined as follows:

$$S \mid\longrightarrow T = \tau(S)$$

where T is defined inside the structure:

$$S \oplus S \oplus \ldots \oplus S$$

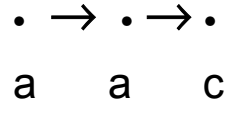
(fixed number of disjoint copies of S)
by MS formulas.

Example of an MS-transduction

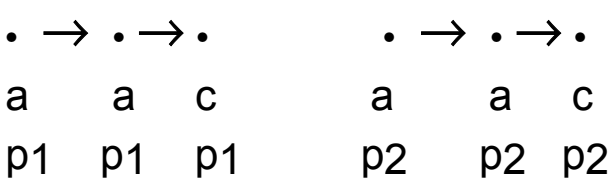
The *square* mapping δ on words: u $|\rightarrow$ uu

We let u = aac

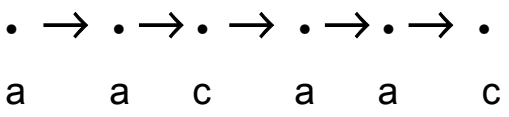
S



S $\oplus$ S



δ(S)



In δ(S) we redefine Suc as follows :

Suc(x,y) : $\Leftrightarrow$ $p_1(x)$ & $p_1(y)$ & Suc(x,y)

v $p_2(x)$ & $p_2(y)$ & Suc(x,y)

v $p_1(x)$ & $p_2(y)$ & "x has no successor"

& "y has no predecessor"

We also remove the "marker" predicates p_1, p_2.

Context-Free Graph Grammars

For words the set of context-free rules:

$$S \rightarrow a\,S\,T \qquad\qquad S \rightarrow b$$

$$T \rightarrow c\,T\,T\,T \qquad\qquad T \rightarrow a$$

is equivalent to the system of set equations:

$$S = a\,S\,T \cup \{ b \}$$

$$T = c\,T\,T\,T \cup \{ a \}$$

where S is the language generated by S (idem for T and T).

For graphs we consider similarily systems of equations like:

$$S = f(k(S), T) \cup \{ b \}$$

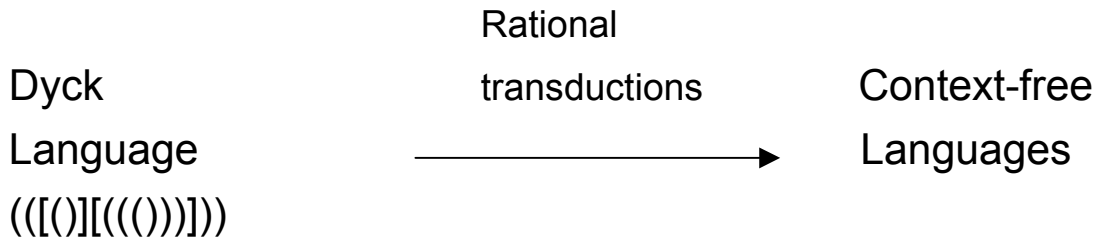
$$T = f(T, f(g(T), m(T))) \cup \{ a \}$$

where f is a binary operation, g, k, m are unary operations on graphs, a, b are basic graphs.

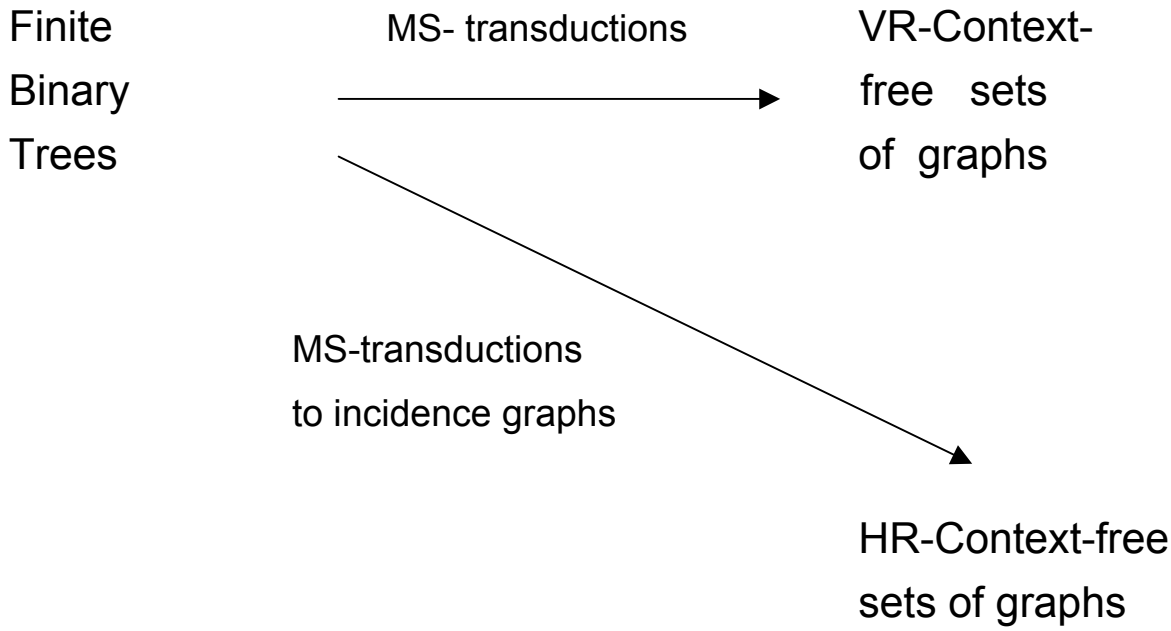
There are two sets of graph operations, related to tree-width and to clique-width, hence we have two classes of context-free sets of graphs, called for historical reasons: HR-context-free (for Hyperedge-Replacement) and VR-context-free (for Vertex-Replacement).

Trees, MS-Transductions and Context-free graph grammars

For words:



For graphs:



Fact: Rational and MS-transductions are closed under composition.

MS-compatible graph transformations.

Definition : Transformations of relational structures S
(or of graphs represented by structures):

$$S \longmapsto \dot{\tau}(S)$$

$$\tau^{\#}(\psi) \longleftarrow \psi$$

such that every MS formula ψ has an effectively computable *backwards translation* $\tau^{\#}(\psi)$, an MS formula, such that :

$$S \models \tau^{\#}(\psi) \quad \text{iff} \quad \tau(S) \models \psi$$

The verification of ψ in the object structure $\dot{\tau}(S)$ reduces to the verification of $\tau^{\#}(\psi)$ in the given structure S.

Informally S describes $\dot{\tau}(S)$ and the MS properties of $\dot{\tau}(S)$ are described by MS properties of S.

Consequence: If a set of structures L has a decidable MS satisfiability problem, then so has $\dot{\tau}(L)$.

Proposition: Every MS-transduction is MS-compatible.

Fact : The semantical mappings of the examples 1 to 5 are

MS-compatible but are not MS-transductions

Theorem (Courcelle, Walukiewicz) : The unfolding mapping from directed graphs to trees is MS compatible.

Consequences:

1. The MS properties of the behaviour of a "program" M (*dynamic properties* of M) can be expressed by MS formulas in M itself (as MS *static* properties of M).

(The translation mapping is the same for all "programs" or "transition systems", i.e., all graphs with same set of edge labels.)

2. The MS satisfiability problem (equiv. the MS theory) of each regular tree is decidable (known from Rabin, 1969) and moreover, reduces to that of the finite graphs of which they are unfoldings.

The recursive mapping : $G \longrightarrow Unf(G) \longrightarrow Th_{MS}(Unf(G))$

factors as : $Th_{MS}(G)$

Example showing that this is not a trivial fact.

Consider the 'shuffle' mapping $\varpi : a^*b^* \longrightarrow \{a,b\}^*$ defined as :

$\varpi(a^n,b^m) = (ab)^n b^{(m-n)}$ if $n \leq m$

$\varpi(a^n,b^m) = (ab)^n a^{(n-m)}$ if $n > m$.

Each output word has a decidable MS-satisfiability problem (trivially since each output word is finite).

However, ϖ is not MS compatible, otherwise the langage a^nb^n would be regular. (Classical argument).

Hence, the MS theory of each word ϖ (w) is decidable but in a way that is *not uniform in terms* of w, considered as a description of ϖ(w).

Observations:

1.The monadic second-order theory of a regular tree is decidable.

2. The (unfolding) mapping from:

Finite Regular Systems $\longrightarrow$ Regular trees

is MS-compatible.

Hence: 2 $\Rightarrow$ 1.

But 2 is much richer. See example of ϖ.

3. MS compatible structure transformations

MS compatible transformations of structures :

1. Each MS-transduction

2. The composition of two MS compatible transformations (clear from the definition)

3. Unfolding

4. The Shelah-Stupp-Muchnik tree expansion (proved by Walukiewicz).

The Shelah-Stupp-Muchnik Construction

Structure M $\longmapsto$ Structure Tree(M)

M = <V, edg(.,.)>

Tree(M) = <Seq(V), edg^{Seq}(.,.),son(.,.),clone(.)>

Seq(V) = nonempty sequences over V,

edg^{Seq} = { (wx,wy) / $w \in V^*$,(x,y) **2** edg}

son = { (w,wx) / $w \in$ Seq(V), x **2** V},

clone = { wxx / $w \in V^*$, $x \in$ V},

(clone = copy of its father).

Theorem (Walukiewicz): The mapping M $\longmapsto$ Tree(M) is MS-compatible.

Observation: For a graph G, the tree Unf(G) is *MS- definable inside* Tree(G) (whence, definable from Tree(G) by an MS-transduction).

Consequence: The mapping Unf is MS-compatible, as composition of two MS-compatible mappings.

Actually the case of unfolding (provable without this difficult theorem) would suffice for the evaluation of first-order substitutions from which follow the cases of algebraic and hyperalgebraic trees.

Conjecture (Seese): If a set of (finite) graphs L has a decidable MS theory, then it is of the form:

$L = \tau(T)$ for T a set of finite trees,
τ an MS transduction.

About the *structure of sets* of graphs having a decidable MS theory.

Related question: What is the *structure of MS-compatible graph transformations* ?

The best we know are transformations of the form:

$\tau = \alpha$ o Tree o β o Tree o γ o..... o Tree o ω

where Tree is the Shelah-et al. construction and α, β ,..., ω are MS transductions. What else ?

4. First-Order substitution as a basic operation

F: function symbols

X: finite set of first-order variables

$T^{\omega}(F,X)$ = finite and infinite terms
(viz. trees) over F and X.

New operation:
for *w* a sequence of *n* pairwise distinct variables in X,

$sub_w(s, t_1, ..., t_n)$ denotes
the result of the substitution in
s of t_i for each occurrence of
the *i*-th variable of *w*.

Sub_X is the (finite) set of all operations sub_w.

We let Eval:

$$T^{\omega}(F \cup Sub_X, X) \longrightarrow T^{\omega}(F,X)$$

be the mapping that *evaluates* the substitution operations

hence eliminates the substitution symbols.

Example: (A, B, C are any finite or infinite terms)

$Eval[sub_{x,y,z}(f(f(x,y),g(x,u)),A,B,C)] = f(f(A,B),g(A,u))$

Theorem (B.C.,T.Knapik): The mapping Eval is MS-compatible.

Proof sketch:

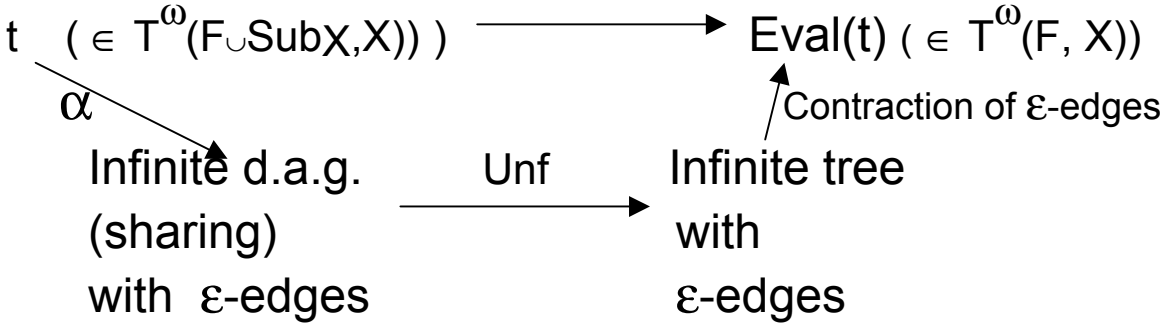
The mapping Eval is the composition

of three MS-compatible mappings:

1) An MS-transduction from terms to directed acyclic graphs,

2) the unfolding Unf,

3) the simultaneous contraction of all ε-labelled edges, also an MS-transduction.



Algebraic trees as Eval(Regular Trees)

A *regular tree* (term) t over F is defined by a finite system like:

$$t = f(t, g(t, s))$$
$$s = g(a, f(s,t))$$

Theorem (Rabin, 1969): MS properties of regular trees are decidable.

An *algebraic tree* (term) over $F \cup X$ can be defined by a recursive applicative program scheme like :

$$\theta(x,y) = f(x, y, \theta(x,g(y)))$$

or by equation (*) over $F \cup Sub_X$ (where substitution operations are evaluated) :

$$t = f(x,y, sub_{x,y}(t, x, g(y))) \qquad (*)$$

Note that t is in argument position. Hence, equation (*) defines a regular tree T over $F \cup Sub_X$ and $t = Eval(T)$.

Theorem:1. $Alg(F,X) = Eval(Reg(F \cup Sub_X, X))$

2.The MS properties of algebraic trees are decidable.

3. The *semantical mapping* from :

Systems(F,X) to Alg(F,X)

is MS compatible.

Remark : Systems(F,X) is the set of (finite) systems of algebraic equations written over the finite sets F and X with unbounded number of ‘unknown’ functions of maximal arity Card(X)

5. Hyperalgebraic trees

We can define a hierarchy of sets of infinite trees:

$Alg^0(F,X) = Reg(F,X)$,

$Alg^1(F,X) = Eval(Reg(F \cup Sub_Y, Y)) \cap T^{\omega}(F,X)$,

...

$Alg^{n+1}(F,X) = Eval(Alg^n(F \cup Sub_Y, Y)) \cap T^{\omega}(F,X)$,.

Theorem: 1. The MS properties of hyperalgebraic trees are decidable.

2. The mapping from systems to theories of the corresponding trees is recursive.

Question : Which types of program scheme do these infinite hyperalgebraic trees represent?

Lambda schemes

Example :

$x,y,u,v,k : b$, $\varphi,\psi,g,h : b \to b$

$f : b \times b \to b$ $H : (b \to b) \times (b \to b) \times b \times b \to b$

$$H(\varphi,\psi,x,y) = f\,[\,x,\ \varphi(H(\ \lambda u.\varphi(\psi(u)),\ \lambda v.f(\varphi(v),\psi(h(v))),\ g(y),\ \varphi(k))\ ]$$

We let $K(\varphi,\psi) = \lambda x,y.H(\varphi,\psi,x,y)$

of type : $(b \to b) \times (b \to b) \to (b \times b \to b)$

K is definable recursively by:

$$K(\varphi,\psi) = \mathrm{comp}_2(f,\ \pi_1,\ \mathrm{comp}_1(\varphi,\ \mathrm{comp}_2(K[\mathrm{comp}_1(\varphi,\psi),\ \mathrm{comp}_2(f,\ \varphi,\mathrm{comp}_1(\psi,h))],\ \mathrm{comp}_1(g,\pi_2),\ \mathrm{comp}_1(\varphi,k_2))))$$

where:

$comp_2(\varphi,\psi,\kappa)$ defined as $\lambda w.\ \varphi(\psi(w), \kappa(w))$

corresponds to $sub_{x,y}(.,.,.)$ and

$comp_1(\varphi,\psi) = \lambda w.\ \varphi(\psi(w))$ corresponds to $sub_x(.,.)$

π_i is the i-th projection : $b \times b \rightarrow b$,

k_2 is the constant mapping = k, with 2 arguments.

Hence K defines an algebraic tree T over $comp_2$, $comp_1$, the nullary symbols f, g, h, π_1 , π_2 , k_2

and H defines an infinite tree t = Eval(T).

Hence t is in Alg^2.

A more difficult case:

$$H(\varphi, x) = f(x, \varphi(H(\lambda u.f(\varphi(u), x), x)))$$

It defines actually an algebraic tree.

Question: Does every lambda-scheme define a hyperalgebraic tree?

An open question:

For a Noetherian and confluent term rewriting system, the normal form mapping goes from finite terms to finite terms .

When is it an MS-transduction ?

When is it MS-compatible?