

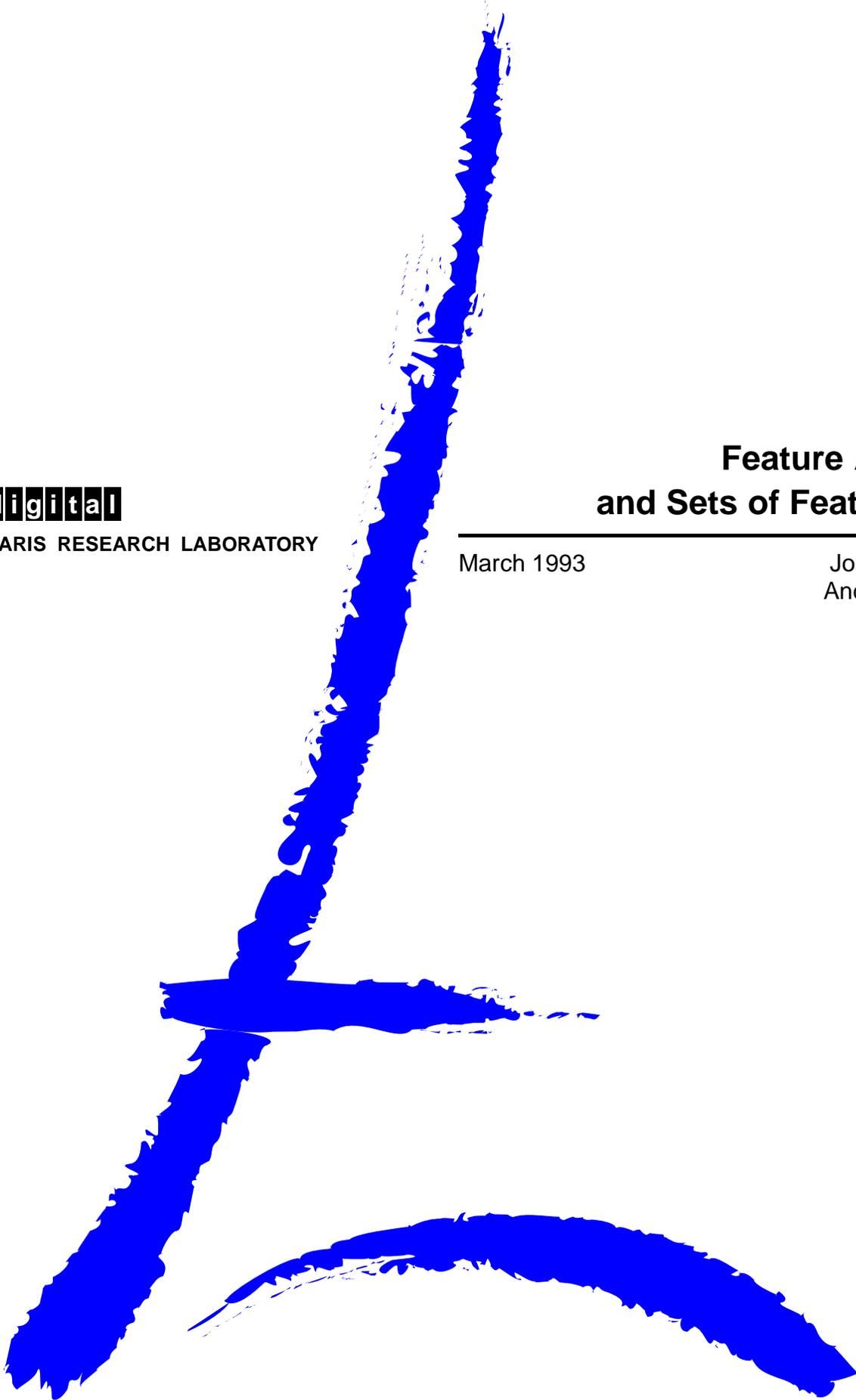
digital

PARIS RESEARCH LABORATORY

Feature Automata and Sets of Feature Trees

March 1993

Joachim Niehren
Andreas Podelski



**Feature Automata
and Sets of Feature Trees**

Joachim Niehren
Andreas Podelski

March 1993

Publication Notes

This report is a revised version of a paper that was presented at the 4th International Joint Conference on the Theory and Practice of Software Development (TAPSOFT'93) in Orsay, France, April 13-16, 1993, and appeared in the proceedings of the conference, edited by Marie-Claude Gaudel and Jean-Pierre Jouannaud, as volume 668 of Springer Lecture Notes in Computer Science 668 (1993), on pages 356-375.

The authors can be contacted at the following addresses:

Joachim Niehren	Andreas Podelski
German Research Center for	Digital Equipment Corporation
Artificial Intelligence (DFKI)	Paris Research Laboratory
Stuhlsatzenhausweg 3	85, avenue Victor Hugo
6600 Saarbrücken 11	92563 Rueil-Malmaison Cedex
Germany	France
niehren@dfki.uni-sb.de	podelski@prl.dec.com

© Digital Equipment Corporation 1993

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for non-profit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of the Paris Research Laboratory of Digital Equipment Centre Technique Europe, in Rueil-Malmaison, France; an acknowledgement of the authors and individual contributors to the work; and all applicable portions of the copyright notice. Copying, reproducing, or republishing for any other purpose shall require a license with payment of fee to the Paris Research Laboratory. All rights reserved.

Abstract

Feature trees generalize first-order trees (which are called ground terms in the general framework of universal algebra). Namely, argument positions become keywords (“features”) from an infinite symbol set \mathcal{F} . A constructor symbol becomes a node symbol that can occur with arbitrary and arbitrarily many argument positions. Feature trees are used to model flexible records; the assumption that \mathcal{F} is infinite accounts for dynamic record field additions.

We develop a universal algebra framework for feature trees. We extend the classical set-defining notions: automata, regular expressions and equational systems, and show that they coincide. This extension of the regular theory of trees requires new notions and proofs. Roughly, a feature automaton reads a feature tree in two directions: along its branches and along the list of the direct descendants of each node. The second direction corresponds to an automaton on a commutative monoid (over an infinite alphabet).

One motivation for this work stems from the fact that, in a type system for the programming language LIFE, the types denote sets of feature trees. Operations needed for type checking can now be implemented by the corresponding automata algorithms.

Résumé

Des arbres à traits généralisent des arbres du premier ordre (qui sont appelés des termes clos dans l’algèbre universelle). À savoir, les positions d’arguments deviennent des mots clés (“traits”) appartenants à un ensemble infini de symboles, \mathcal{F} . Un symbole de constructeur devient un symbole de noeud qui peut apparaître avec n’importe quelles positions d’arguments en n’importe quel nombre. Des arbres à traits sont utilisés pour la modélisation des enregistrements flexibles; la supposition de l’infinitude de \mathcal{F} est nécessaire pour rendre compte des additions dynamiques de champs d’enregistrements.

Nous développons un cadre formel pour les arbres à traits, dans l’algèbre universelle. Nous étendons les notions classiques définissant les ensembles : les automates, les expressions régulières et les systèmes équationnels, et nous montrons qu’elles coïncident. Cette extension de la théorie régulière des arbres nécessite de nouvelles notions et preuves. Schématiquement, un automate à traits lit un arbre à traits dans deux directions : le long de ses branches, et le long de la liste des descendants directs de chaque noeud. Cette deuxième direction correspond à un automate sur un monoid commutatif.

Une motivation de ce travail vient du fait que, dans un système de types pour le langage de programmation LIFE, les types dénotent des ensembles d’arbres à traits. Des opérations utilisées pour la vérification de types peuvent maintenant être implémentées par les algorithmes d’automates correspondants.

Keywords

Logic programming, types, constraint systems, constructor trees, feature logic, feature trees, tree automata.

Acknowledgements

The authors are grateful to Hassan Aït-Kaci and to Gert Smolka for first arousing their individual interest in the idea of using tree automata for feature constraint solving and then bringing them together. They also wish to thank Bruno Dumant, Hubert Comon, Helmut Seidl and Ralf Treinen for encouraging discussions. Finally, the anonymous referees of TAPSOFT'93 provided helpful and intriguing comments.

Contents

1	Introduction	1
2	The Algebra \mathcal{J}	3
3	Feature Automata	5
4	Counting Constraints	7
5	Kleene's Theorem	9
6	Equational Systems	10
7	Conclusion and Further Work	11
A	The Algebra of Multisets	12
B	Counting in Multisets	16
	References	19

1 Introduction

In this section, we will give some background and motivation (“the task”) and then outline the rest of the paper (“the method”).

The Task. We describe a specific formalism of data structures called feature trees. They are a generalization of first-order trees, also called constructor trees or the elements of the Herbrand universe. Since trees have been useful, *e.g.*, for structuring data in modern symbolic programming languages like Prolog and ML, the more flexible feature trees have an interesting potential. Precisely, feature trees model record structures. They form the semantics of record calculi like [AK86], which are used in programming languages [AKP91b] and in computational linguistics (*cf.*, the book [Car92]). In the logical framework for record structures of [AKPS92], they constitute the interpretation of a first-order theory, which is completely axiomatizable, and hence decidable [BS92].

As graphs, feature trees are easily described as finite trees whose nodes are labeled by node symbols (instead of constructor symbols), and whose edges are labeled by feature symbols (instead of being numbered), all those edges outgoing from the same node by different ones. Thus, symbolic keywords called features denote the possible argument positions of a node. They access uniquely the node’s direct subtrees. All node symbols can label a node with any features attached to it, in any, though finite, number.

Although thoroughly investigated [AK86, Smo92, BS92, AKPS92], also in comparison with first-order trees [ST92], feature trees have never been characterized as composable elements in an algebraic structure, *i.e.*, with operations defined on them. Also, up to now, there has been no corresponding notion of automaton. This device has generally proven useful for systems calculating over sets of elements.

The practical motivation for such a system comes from the possibility of defining a hierarchy of types denoting sets of feature trees. For its use in a logical programming system employing feature trees such as LIFE [AKP91b], we need to compute efficiently the intersection of two types (roughly, for unification). Concurrent systems, in connection with control mechanisms such as residuation or guards [AKP91a], require furthermore an efficient test of the subset relation (matching). Thus, we need to provide a formalism defining the types in a way that is expressive enough and yet keeps the two problems decidable. Such a formalism can be given, for example, as a system of equations and the corresponding automata notion with Boolean closure properties and decidable emptiness problem.

Also, if we want to extend the techniques of type systems for logic programming, where types denote sets of trees (*cf.*, the book [Pfe92]), to LIFE, where types will instead denote sets of feature trees, we first have to provide a corresponding formal framework.

A major difficulty in the construction of a suitable algebraic framework for feature trees (*i.e.*, with the property that automata and equational systems coincide¹) comes from the fact that the set \mathcal{F} of features, *i.e.*, of possible argument positions of a node accessing its direct subtrees, is

¹We note that the expressiveness of tree automata is equal to the one of equational systems for the free term algebras over finite signatures; it is strictly weaker in the case of infinite signatures for all tree species, including those considered in [Cou89, Cou92].

infinite. The infiniteness of \mathcal{F} is, however, an essential ingredient of the formal frameworks modeling structures. A practical motivation of the infiniteness is the need to account for the possibility of dynamic addition of (arbitrarily many) record fields to a value. It turns out that this semantical point of view has advantages for implementation as well. Namely, the correctness of the efficient algorithms for entailment and for solving negated constraints on feature trees [AKPS92] relies on the infiniteness of \mathcal{F} .

The Method. The first step in solving the problem described above is to build an appropriate algebraic framework. Such a framework is provided by universal algebra in the case of first-order trees. Formally, these are the elements of the free algebra over a given signature of function symbols (finite or infinite, *cf.*, [Mah88]). This framework yields immediately a “good” notion of automata.

In fact, as Courcelle has shown in [Cou89, Cou92], universal algebra provides a framework for a rich variety of trees. Clearly, that work inspired our notion of the algebra underlying feature trees. We introduce this notion in Section 2. Informally speaking, the operation composing feature trees in the algebra takes a record value and adds a record field containing another value to it. In a special case, this amounts to Nivat’s notion of ‘sum of trees’ [Niv92]; thus, incidentally, we obtain an algebraic formalization hereof.

To define feature automata as algebras, it is useful to consider the class of all finite trees whose nodes are labeled by node symbols, and whose edges are labeled by feature symbols. We call these multitrees.² Multitrees are of interest on their own, namely for representation of knowledge with set-valued attributes [Rou88]. Thus, feature trees are multitrees with the restriction that features are “functional,” *i.e.*, all edges outgoing from the same node are labeled with different features. Feature automata recognize languages of multitrees, which are then cut down to recognize languages of feature trees.

In Section 3, we will define feature automata and show the basic properties of this notion: closure under the Boolean operations and decidability of the emptiness problem. In order to restrict our study to finitely representable automata and yet to account for the infiniteness of the set of features \mathcal{F} , we introduce the notion of a **finitary automaton**: the number of states is finite, and the evaluation of the automaton can be specified not only on single symbols, but also on finite sets or on complements of finite sets of symbols. Thus, it could be specified by saying either “the value of $f \dots$ for all symbols $f \in F$ ” or “the value of $f \dots$ for all symbols $f \notin F$,” where $F \subseteq \mathcal{F}$ is finite.

Roughly, a feature automaton reads a feature tree in two directions: along its branches (from the frontier to the root) and along the fan-out of each node (along all argument positions). This is necessary in order to account for the flexibility in the depth as well as in the out-degree of the nodes of feature trees. The first direction is standard for all automata over trees. In order to study its behavior in the latter direction, or what we call the local structure of the recognized language, we consider recognizable sets of feature trees of depth 1, called flat feature trees.

²The unranked unordered trees studied in [Cou89] (the number of arguments of the nodes is not restricted, and the arguments are not ordered) are a special case of multitrees, namely with just one feature. In the framework of [Cou89], however, recognizability by automata is strictly weaker than definability by equational systems, even if the set of node labels is finite.

In Section 4, we define a class of logical formulas, called **counting constraints**. The name comes from the fact that they express threshold or modulo counting of the subtrees which are accessed via features from a finite or co-finite set of features. That is, their occurrences are counted up to a certain number, or modulo a certain number.

The main technical result of this paper is a theorem saying that counting constraints characterize exactly the recognizable sets of flat feature trees. The proof takes up Sections A and B. The theorem essentially links counting and the finitary-condition; in all of the set-defining devices presented here, either of these two notions accounts for the infiniteness of \mathcal{F} .

Counting constraints can express that certain features exist in the flat feature tree (labeling edges from the root), and that others do not.³ As a consequence, one can show that the set of first-order trees, with fixed arity assigned to node symbols, and recognizable subsets of these, are sets recognized by feature automata.

In Sections 5 and 6, we give two alternative ways to define recognizable sets of feature trees which are more practical than automata: regular expressions and equational systems. In the first one, the sets are constructed by union, substitution and star, *i.e.*, iterated substitution (and, optionally, complement or intersection). In the second, they are defined as solutions of equations in a certain form. For both, counting constraints can be used to define the base cases. Thanks to the main theorem in Section 4, we are able to show that either class of defined sets is equal to the one for feature automata. Moreover, the devices can be effectively translated one into the other. These results, together with the previous ones, are necessary to present a complete regular theory of feature trees and to offer a solution to the practical problem of computing with types denoting sets of feature trees as described above.

2 The Algebra \mathcal{J}

In this section, we will introduce feature trees and the more general multitrees as elements of an algebra that we define, called \mathcal{J} . This yields the notion of a \mathcal{J} -automaton. This section follows the approach of [Cou89] and [Cou92].

In the following we will assume a given set \mathcal{S} of node symbols⁴ (referred to by $A, B, \text{etc.}$) and a given set \mathcal{F} of feature symbols (also called attributes, or record field selectors, referred to by $f, g, \text{etc.}$).

Formally, **multitrees** are trees (*i.e.*, finite directed acyclic rooted graphs) whose nodes are labeled over \mathcal{S} , and whose edges are labeled over \mathcal{F} . Or, the set \mathcal{MT} of multitrees over \mathcal{S} and \mathcal{F} can be introduced as $\mathcal{MT} = \bigcup_{n \geq 0} \mathcal{MT}_n$ where (let \mathbf{N} denote the set of all natural numbers, and \mathbf{N}_{finite}^M the set of finite multisets with elements from the set M):

$$\begin{aligned} \mathcal{MT}_0 &= \{(A, \emptyset) \mid A \in \mathcal{S}\}, \\ \mathcal{MT}_n &= \{(A, E) \mid A \in \mathcal{S}, E \in \mathbf{N}_{finite}^{\mathcal{F}} \times \mathcal{MT}_{n-1}\} \cup \mathcal{MT}_{n-1}. \end{aligned}$$

³In [ST92, Smo92], these correspond to the constraints $xF, xf \downarrow$ or their negations, where $F \subseteq \mathcal{F}$ finite and $f \in \mathcal{F}$.

⁴In the literature on feature trees, the elements of \mathcal{S} are usually called “sorts.” In this text, we use “node symbols” instead of “sorts” in order to avoid confusion with the notion of sorts of domains in universal algebra.

\mathcal{MT}_n contains the multitrees of depth $\leq n$.

Feature trees are multitrees such that all edges outgoing from the same node are labeled by different features. \mathcal{FT} denote the set of all feature trees (and \mathcal{FT}_n all those of depth $\leq n$).

We introduce two sorts MT and F for multitrees and features, respectively, and define the $\{MT, F\}$ -sorted signature:

$$\Sigma = \{\Rightarrow\} \uplus \mathcal{F} \uplus \mathcal{S}$$

where \Rightarrow is a function symbol of profile: $MT \times F \times MT \mapsto MT$, and the symbols in \mathcal{F} and \mathcal{S} are constants of sort F and of sort MT , respectively.

The **algebra of multitrees** \mathcal{J} is defined as a Σ -algebra. Its two domains are $D_{MT} = \mathcal{MT}$ and $D_F = \mathcal{F}$ of the sorts MT and F , respectively. Its ternary function symbol $\Rightarrow^{\mathcal{J}}$ is interpreted in \mathcal{J} as the operation which composes two multitrees $t, t' \in \mathcal{MT}$ via a feature $f \in \mathcal{F}$ to a new multitree composed of t and t' with an edge labeled f from the root of t to the root of t' . Or (where \sqcup denotes multiset union),

$$\Rightarrow^{\mathcal{J}}((A, E), f, t) = (A, E \sqcup \{(f, t)\}).$$

Borrowing the ‘tree sum’ notation from [Niv92], we might write $\Rightarrow^{\mathcal{J}}(t, f, t')$ more intuitively as $t + ft'$. In fact, for the special case where $\mathcal{F} = \{1, 2\}$ (the two features denoting left and right successors), we obtain an algebraic reading of the notation of [Niv92].

The interpretation of the constants is given by $f^{\mathcal{J}} = f$ and $A^{\mathcal{J}} = (A, \emptyset)$.

It is easy to verify that the algebra \mathcal{J} satisfies the **order independence theory (OIT)**, *i.e.*, the following equation is valid in \mathcal{J} .

$$\Rightarrow(\Rightarrow(x, f_1, x_1), f_2, x_2) = \Rightarrow(\Rightarrow(x, f_2, x_2), f_1, x_1) \quad (1)$$

In the ‘tree sum’ notation this expresses the commutativity⁶ of $+$, in the sense that $t + f_1 t_1 + f_2 t_2 = t + f_2 t_2 + f_1 t_1$. Of course, always $t + f_1 t_1 + f_2 t_2 \neq t + f_1(t_1 + f_2 t_2)$.

We use \mathcal{T}_{Σ} to denote the free algebra of terms over the signature Σ .

Lemma 1 *The algebra of multitrees \mathcal{J} is isomorphic to the quotient of the free term algebra over Σ with the least congruence generated by the order-independence equation (1),*

$$\mathcal{J} = \mathcal{T}_{\Sigma}/\text{OIT}.$$

We note the well-known fact that, given any system of equations \mathcal{E} , $\mathcal{T}_{\Sigma}/\mathcal{E}$ is the initial object in the category of all Σ -algebras satisfying the equations \mathcal{E} .

A \mathcal{J} -automaton is a tuple $(\mathcal{A}, h, Q_{\text{final}})$ consisting of a finite Σ -algebra \mathcal{A} , a homomorphism $h : \mathcal{J} \mapsto \mathcal{A}$ and the subset $Q_{\text{final}} \subseteq D_{MT}^{\mathcal{A}}$ of values of sort MT (“final states”) where the number of values of sort MT and of sort F (“states”) is finite. A \mathcal{J} -automaton corresponds to the “more

⁵We use the symbol \Rightarrow in reminiscence of the notation for record descriptions like ψ -terms in [AK86, AKP91b], which are of the form $\psi = X : s(f_1 \Rightarrow \psi_1, \dots, f_n \Rightarrow \psi_n)$.

⁶In a sense which can be made formal (*cf.*, Section A), also the associativity holds for $+$; this justifies dropping the parenthesis.

concrete” notion of a (finite deterministic bottom-up) tree automaton over the terms of T_{Σ} such that all terms which are equal modulo OIT are evaluated to the same state. This means that any representation of a multitree t as a term in T_{Σ} can be chosen in order to calculate the value of t .

3 Feature Automata

Given any many-sorted signature Σ with a finite number of non-constant function symbols $c \in (\Sigma - \Sigma_s^0)$ for every sort s , we define a Σ -algebra \mathcal{A} to be **finitary** if, for each sort s and each value $q \in D_s^{\mathcal{A}}$ of sort s , the set:

$$\{c \in \Sigma_s^0 \mid c^{\mathcal{A}} = q\}$$

of constant symbols in Σ of sort s which are valued to q is finite or co-finite.

We now return to the particular $\{MT, F\}$ -sorted signature Σ introduced above; clearly, the definitions below can be made in the general framework as well.⁷

A **feature automaton** \mathcal{A} is defined as a finitary \mathcal{J} -automaton. The set of multitrees recognized by \mathcal{A} is the set:

$$L_{\mathcal{MT}}(\mathcal{A}) = \{t \in \mathcal{MT} \mid h(t) \in Q_{\text{final}}\},$$

and the set of feature trees recognized by \mathcal{A} is the set: $L_{\mathcal{FT}}(\mathcal{A}) = L_{\mathcal{MT}}(\mathcal{A}) \cap \mathcal{FT}$. The families $Rec_{\mathcal{MT}}(\mathcal{J})$ and $Rec_{\mathcal{FT}}(\mathcal{J})$ of **recognizable sets** of multitrees and feature trees are defined accordingly.

Remark. If (and only if) the set of features is infinite, the set of all feature trees is not a recognizable language of multitrees (with respect to \mathcal{J}).

Example. We will construct a feature automaton \mathcal{A} that recognizes the set of natural numbers. These are coded into the feature trees of the form $(0, \{(succ, (0, \{(succ, (\dots, \{(0, \emptyset)\})\})\})\})$, with n edges labeled *succ* for the natural number n . As elements in the quotient term algebra T_{Σ}/OIT , they would be written as the singleton congruence classes $\{\Rightarrow (0, succ, \Rightarrow (0, succ, \Rightarrow (\dots, 0)))\}$. The feature automaton \mathcal{A} has the states $Q = \{q_{\text{nat}}, q_{\text{other}}\}$ and $P = \{p_{\text{succ}}, p_{\text{other}}\}$ of sort *MT* and *F*, respectively. The evaluation is given by:

$$\begin{aligned} 0^{\mathcal{A}} &= q_{\text{nat}}, \\ A^{\mathcal{A}} &= q_{\text{other}} \quad \text{if } A \neq 0, \\ succ^{\mathcal{A}} &= p_{\text{succ}}, \\ f^{\mathcal{A}} &= p_{\text{other}} \quad \text{if } f \neq succ, \\ \Rightarrow^{\mathcal{A}}(q_{\text{nat}}, p_{\text{succ}}, q_{\text{nat}}) &= q_{\text{nat}}, \\ \Rightarrow^{\mathcal{A}}(q_1, p, q_2) &= q_{\text{other}} \quad \text{otherwise.} \end{aligned}$$

As final state set we choose $Q_{\text{final}} = \{q_{\text{nat}}\}$. It is clear that \mathcal{A} respects the order independence theory and the finitary-condition. Of course, it will be more practical to define this set by regular expressions or equational systems.

⁷Also, the finitary-condition: finite or co-finite, could be made more general such that the proof of Theorem 1 still holds.

The following theorem and corollary state that the standard properties of recognizable languages are valid for the sets in $Rec_{\mathcal{FT}}$ as well.

Theorem 1

1. *The family of recognizable languages of feature trees $Rec_{\mathcal{FT}}$ is closed under the Boolean operations. The corresponding feature automata can be given effectively.*
2. *The emptiness problem ($L_{\mathcal{FT}}(\mathcal{A}) \stackrel{?}{=} \emptyset$) is decidable for each feature automaton \mathcal{A} .*

Proof. The known constructions for Boolean operations on automata are still valid for \mathcal{J} -automata. To see that the finitary-condition is preserved, simply note that the system of finite and co-finite sets is Boolean closed and, for two states q_1 and q_2 of the feature automata \mathcal{A}_1 and \mathcal{A}_2 , respectively,

$$\{c \in \Sigma_s^0 \mid c^{(\mathcal{A}_1, \mathcal{A}_2)} = (q_1, q_2)\} = \{c \in \Sigma_s^0 \mid c^{\mathcal{A}_1} = q_1\} \cap \{c \in \Sigma_s^0 \mid c^{\mathcal{A}_2} = q_2\}.$$

Since $\mathcal{J} = \mathcal{T}_{\Sigma/\text{OIT}}$, each \mathcal{J} -automaton \mathcal{A} corresponds to a tree automaton \mathcal{A}_T over terms in \mathcal{T}_{Σ} , and:

$$L_{\mathcal{FT}}(\mathcal{A}) = \emptyset \text{ iff } L_{T_{\Sigma}}(\mathcal{A}_T) = \emptyset,$$

it suffices to decide the emptiness problem for the tree automaton \mathcal{A}_T . As usual, this can be done by checking all terms of depth smaller than the number of states of \mathcal{A}_T . Let C be some finite set of constants c such that $c^{\mathcal{A}} = q$ for each state q which is a value of some constant. If (and only if) L is not empty, it contains a term of bounded depth that is constructed with constants of C and non-constant function symbols. But there are only finitely many terms of this kind.

A finitary automaton can be finitely represented. From such a representation one can calculate some set C as described above. This yields an algorithm for testing $L_{\mathcal{MT}}(\mathcal{A}) = \emptyset$. In the case of $L_{\mathcal{FT}}(\mathcal{A})$ the algorithm checks only terms representing feature trees. \square

We conclude the section by defining non-deterministic feature automata which are needed in Sections 5 and 6.

Definition 1 A **non-deterministic feature automaton** $\mathcal{A} = (Q, P, h, Q_{\text{final}})$ is a tuple such that:

Q is the set of states of sort MT , P the set of states of sort F and $Q_{\text{final}} \subseteq Q$ the set of final states,

h is composed of the functions $h : \mathcal{S} \rightarrow 2^Q$ and $h : \mathcal{F} \rightarrow 2^P$ and the transition function $\Rightarrow^{\mathcal{A}} : Q \times P \times Q \rightarrow 2^Q$,

\mathcal{A} satisfies OIT, i.e., for all states q, p_1, q_1, p_2, q_2 ,

$$\Rightarrow^{\mathcal{A}}(\Rightarrow^{\mathcal{A}}(q, p_1, q_1), p_2, q_2) = \Rightarrow^{\mathcal{A}}(\Rightarrow^{\mathcal{A}}(q, p_2, q_2), p_1, q_1),$$

and \mathcal{A} satisfies the finitary-condition, i.e., for all states p and q , the sets $\{f \in \mathcal{F} \mid p \in f^{\mathcal{A}}\}$ and $\{A \in \mathcal{S} \mid q \in A^{\mathcal{A}}\}$ are finite or co-finite.

The evaluation of the term $t \in \mathcal{T}_{\Sigma}$ by \mathcal{A} , *i.e.*, the set $h(t) \subseteq Q$ is defined inductively by:

$$h(\Rightarrow (t_1, f, t_2)) = \Rightarrow^{\mathcal{A}} (h(t_1), h(f), h(t_2)).$$

If t_1 and t_2 are congruent modulo OIT, we have $h(t_1) = h(t_2)$. Thus, $h([t]) = h(t)$ is well defined for all congruence classes $[t]$. The language of multitrees recognized by \mathcal{A} is:

$$L_{\mathcal{MT}}(\mathcal{A}) = \{[t] \mid h([t]) \cap Q_{\text{final}} \neq \emptyset\},$$

and the language of feature trees recognized by \mathcal{A} is $L_{\mathcal{FT}}(\mathcal{A}) = L_{\mathcal{MT}}(\mathcal{A}) \cap \mathcal{FT}$. Each feature automaton is also a non-deterministic feature automaton.

Lemma 2 *Given a non-deterministic feature automaton \mathcal{A} , an equivalent (deterministic) feature automaton \mathcal{A}^d can be constructed effectively.*

Proof We apply the usual subset construction on a given non-deterministic feature automaton \mathcal{A} of the form above, yielding the equivalent automaton \mathcal{A}^d as follows: $Q^d = 2^Q$, $P^d = 2^P$, $A^{\mathcal{A}^d} = A^{\mathcal{A}}$, $f^{\mathcal{A}^d} := f^{\mathcal{A}}$, and:

$$\Rightarrow^{\mathcal{A}^d} (q_1^d, p^d, q_2^d) = \bigcup \{ \Rightarrow^{\mathcal{A}} (q_1, p, q_2) \mid (q_1, p, q_2) \in q_1^d \times p^d \times q_2^d \}.$$

We define the final states of \mathcal{A}^d by: $Q_{\text{final}}^d = \{q^d \mid q^d \cap Q_{\text{final}} \neq \emptyset\}$.

Clearly, the algebra \mathcal{A}^d satisfies the OIT-axiom. The equality: The finitary-condition is preserved, since:

$$\{A \mid A^{\mathcal{A}^d} = q^d\} = \bigcap_{q \in q^d} \{A \mid q \in A^{\mathcal{A}}\} \cap \bigcap_{q \notin q^d} \{A \mid q \in A^{\mathcal{A}}\}^C$$

shows that the finitary-condition is preserved, too. □

4 Counting Constraints

In this section we characterize recognizable languages of flat feature trees using formulae of a certain form, called counting constraints. The proof of this characterization, which is the main technical result of this paper, will be done in Sections A and B.

The syntax of **counting constraints** C (written $C(x)$ to indicate that x is the only free variable) is defined in the BNF style as follows (where F is a finite or co-finite sets of features, $n, m \in \mathbf{N}$ are natural numbers, and S is a finite or co-finite subset of \mathcal{S}).

$$C(x) ::= \begin{array}{l} \text{card } \{\varphi \in F \mid \exists y. (x\varphi y \wedge Ty)\} = n \bmod m \\ \mid \\ Sx \\ \mid \\ C(x) \wedge C(x) \\ \mid \\ \neg C(x) \end{array} \quad (2)$$

The counting constraint $C(x) \equiv \text{card } \{\varphi \in F \mid \exists y. (x\varphi y \wedge Ty)\} = n \bmod m$ holds for the multitree x if the number of all edges in x which: (1) go from the root to a node labeled by

a symbol in T and (2) are labeled by a feature φ in F , is equal to $n \bmod m$.⁸ The cardinality operator $card$ applies on a multiset of features, *i.e.*, counts their double occurrences.

The counting constraint $C(x) \equiv Sx$ holds for the multitree x if the root of x is labeled by some symbol in S .

We note the following fact (*cf.*, [Eil74]).

Fact 1 *A language of natural numbers is recognizable iff it can be decomposed into a finite union of sets of the form: $\{n + k \cdot m \mid k \in \mathbb{N}\}$, with $n, m \in \mathbb{N}$.*

Thus, we can define the syntax of counting constraints equivalently in the form (where N is a set of natural numbers which is recognizable in the monoid $(\mathbb{N}, +, 0)$; S , and T , a finite or co-finite subset of \mathcal{S} ; F a finite or co-finite sets of features):

$$C(x) ::= \begin{array}{l} card \{ \varphi \in F \mid \exists y. (x\varphi y \wedge Ty) \} \in N \\ | \\ Sx \\ | \\ C(x) \wedge C(x) \\ | \\ C(x) \vee C(x) \end{array} \quad (3)$$

Note that this definition, too, yields immediately that counting constraints are closed under negation. Indeed, $\neg card \{ \varphi \in F \mid \exists y. (x\varphi y \wedge Ty) \} \in N$ is equivalent to $card \{ \varphi \in F \mid \exists y. (x\varphi y \wedge Ty) \} \in N^c$, and $\neg Tx$ is equivalent to $T^c x$.

Some important feature constraints can be expressed by our new constraints. For example, in the syntax of [Smo92], for $F \subseteq \mathcal{F}$ finite, for $f \in \mathcal{F}$, and for $A \in \mathcal{S}$: xF (“for exactly the features f in F there exists one edge labeled f from the root”), $xf \downarrow$ (“there exists no edge labeled f from the root”), and Ax (“the root is labeled by A ”).

$$xF \equiv \bigwedge_{f \in F} card \{ \varphi \in \{f\} \mid \exists y. x\varphi y \} \in \{1\} \\ \wedge \quad card \{ \varphi \in F^c \mid \exists y. x\varphi y \} \in \{0\},$$

$$xf \downarrow \equiv card \{ \varphi \in \{f\} \mid \exists y. x\varphi y \} \in \{0\},$$

$$Ax \equiv \{A\}x.$$

Each constraint $C(x)$ defines the set $L_{MT}(C)$ of multitrees x for which the constraint $C(x)$ holds. Accordingly, we define: $L_{FT}(C) = L_{MT}(C) \cap \mathcal{FT}$, $L_{MT_1}(C) = L_{MT}(C) \cap \mathcal{MT}_1$, and $L_{FT_1}(C) = L_{FT}(C) \cap \mathcal{FT}_1$. The languages of flat multitrees of the form $L_{MT_1}(C)$, or of flat feature trees $L_{FT_1}(C)$, are called **counting-definable**.

The following theorem holds for multitrees instead of feature trees, as well.

Theorem 2 *A language of flat feature trees is counting-definable iff it is recognizable (in \mathcal{J} , by a feature automaton).*

⁸We define $n \bmod 0 = n$, although this is not quite standard. That is, “counting” means here threshold- and modulo counting.

Proof Sketch. A flat multitree can be represented as a finite multiset over $(\mathcal{F} \cup \{root\}) \times \mathcal{S}$. The operation $\Rightarrow^{\mathcal{J}}$ corresponds to the union of such multisets. In Section A we study the algebra \mathcal{M} of finite multisets of pairs. It is three-sorted, the sorts denoting $\mathcal{F} \cup \{root\}$, \mathcal{S} and \mathcal{MT} , respectively. We show that \mathcal{J} - and \mathcal{M} -recognizability coincide.

In Section B, we consider counting constraints $D(x)$ for multisets x of \mathcal{M} . They are of the form:

$$D(x) \equiv \text{card}\{(f, A) \in x \mid f \in F, A \in T\} \in N,$$

or conjunctions or disjunctions of these. Again F and T are finite or co-finite subsets of \mathcal{F} and \mathcal{S} and N is a recognizable set of natural numbers.

We show that definability of languages of multisets by these constraints and \mathcal{M} -recognizability coincide. The main idea is that the mapping:

$$x \mapsto \text{card}\{(f, A) \in x \mid f \in F, A \in T\}$$

is essentially a homomorphism from \mathcal{M} into \mathbf{N} . □

The theorem above expresses that feature automata can count features either threshold or modulo a natural number.

5 Kleene's Theorem

We define regular expressions over feature trees. In generalization of the standard cases, the atomic constituents of these are not just constants (denoting singletons or trees of depth 1), but expressions which denote sets of feature trees of depth ≤ 1 .

As usual, we need construction variables labeling the nodes where the substitution and the Kleene star operations can take place. These variables are taken from a set Y which is assumed given (disjoint from \mathcal{S}). It is infinite; the definition of each regular language, of course, uses only a finite number of construction variables. We call a syntactic expression C of the form (2) a **counting-expression** if T ranges over finite or co-finite subsets of $\mathcal{S} \cup Y$. Its denotation is defined as the set of all feature trees of depth ≤ 1 which satisfy it as a counting constraint over the extended alphabet of sorts.

A **regular expression** R over \mathcal{F} and $\mathcal{S} \cup Y$ is of the form given by:

$R ::=$	C	C is a counting-expression
	$R \cdot_y R$	concatenation (where $y \in Y$)
	R^{*y}	Kleene star (where $y \in Y$)
	$R \cup R$	union

Complement and intersection are optional operators, which, as we will see, do not properly add expressiveness.

The definition of the language $L_{\mathcal{FT}}(R)$ of feature trees (or $L_{\mathcal{MT}}(R)$ of multitrees) denoted by the regular expression R is by straightforward induction. For concatenation and Kleene star for sets of multitrees: If L_1 and L_2 are sets of feature trees, then $L_1 \cdot_y L_2$ is obtained by replacing the construction variable y in the leaves of the trees of L_1 by (possibly different) trees of L_2 .

The Kleene star operation on a set is an iterated concatenation of a set with itself. Formally, for a set L of feature trees, $L_y^1 = L$, $L_y^n := L_y^{n-1} \cdot_y L$, and $L^{*y} = \bigcup_{n \geq 1} L_y^n$.

The languages of feature trees (or multitrees) denoted by regular expressions are called **regular languages**.

Theorem 3 (Kleene) *A language of feature trees (or multitrees) is regular iff it is recognizable.*

Proof. It is sufficient to prove the theorem for multitrees. We show by induction over the structure of the regular expressions that the language of each regular expression over $\mathcal{S} \cup Y$ and \mathcal{F} is recognizable. The base case $R = C$ is handled by Theorem 2. Union is captured by the Boolean closure properties in Theorem 1. Substitution and star are established using the equivalence of deterministic and non deterministic feature automata. For the other direction, we use the standard McNaughton/Papert induction technique, the base case being handled again by Theorem 2. \square

6 Equational Systems

The next possibility to define recognizable sets of feature trees (or multitrees) in a convenient way uses equational systems. These systems again generalize the constituents from singletons of trees of the form a or $f(y_1, \dots, y_n)$, for $a \in \Sigma_0$ and $f \in \Sigma_n$ in the case of a ranked signature for first-order trees, to counting-expressions denoting (unions of) sets of flat feature trees.

The extra symbols $y \in Y$ in these counting expressions now correspond to set variables of the equations.

We write $C(y_1, \dots, y_n)$ instead of C if the set variables of C are contained in the set $\{y_1, \dots, y_n\}$. These variables are not to be confused with the logical variable x used in $C = C(x)$ as a logical formula.

An **equational system** is a finite set \mathcal{E} of equations of the form (where C_i is a counting-expression, for $i = 1, \dots, n$):

$$y_i = C_i(y_1, \dots, y_n).$$

Given an assignment, *i.e.*, a mapping $\alpha : Y \mapsto 2^{\mathcal{FT}}$, the equations in \mathcal{E} are interpreted such that $C_i(y_1, \dots, y_n)$ denotes the set:

$$L_{\mathcal{FT}}(C_i) \cdot_{y_1} \alpha(y_1) \cdot_{y_2} \dots \cdot_{y_n} \alpha(y_n).$$

A solution of \mathcal{E} is an assignment α satisfying \mathcal{E} . Each equational system has a least solution. The existence follows with the usual fixed point argument. Namely, an equational system is considered as an operator over the lattice of assignments α and the least solution is obtained in ω iteration steps of this operator, starting with the assignment $\alpha(y_i) = \emptyset$ for $i = 1, \dots, n$.

A language of feature trees is called **equational** if it is the union of some of the sets $\alpha(y_i)$ for the least solution α of \mathcal{E} . The notion is defined accordingly for multitrees.

We can now formulate the last characterization of recognizability:

Theorem 4 *A language of feature trees (or multitrees) is equational iff it is recognizable.*

Proof Since \mathcal{J} -recognizability corresponds to the characterization by congruence relations, and Theorem 2 covers the case of feature trees of depth ≤ 1 , the proof can be done following the standard one for first-order trees (*cf.*, [GS84]). \square

7 Conclusion and Further Work

The results of this paper together present a complete regular theory of feature trees. They offer a solution to the concrete practical problem of computing with types denoting sets of feature trees as described in the introduction.

Now, it is interesting to investigate where, in the wide range of applications of first-order trees, feature trees can be useful in replacing or extending those. Since tree automata play a major role, either directly or just by underlying some other formalism, the regular theory of feature trees developed here is a prerequisite for this investigation.

A more speculative application might be conceived as part of the compiler optimizer of the programming language LIFE [AKP91b]. Namely, unary predicates over feature trees defined by Horn clauses without multiple occurrences of variables define recognizable sets of feature trees. Now, satisfiability of the conjunction of two such predicates corresponds to non-emptiness of the intersection of the defined sets. When used in deep guards, entailment of a predicate by others of this kind corresponds to the subset relation on the defined sets of feature trees.

We are curious to extend the developed theory in the following ways. First, we would like to find a logical characterization of the class of recognizable feature trees, extending the results of Doner, Thatcher/Wright and Courcelle [Don70, TW67, Cou90]. It will be interesting to combine second-order logic and the counting constraints introduced here, in order to account for the flexibility in the depth as well as in the out-degree of the nodes of feature trees.

Also, in order to account for circular data structures, like, *e.g.*, circular lists, it is necessary to consider infinite (rational) feature trees. Thus, it would be useful to construct a regular theory of these.

Finally, in [CD91] it is shown that the first-order theory of a tree automaton is decidable (in the case of a finite signature). More precisely, it is possible to solve first-order formulas built up from equalities between first-order terms and membership constraints of the form $x \in q$, where q denotes a set defined by a tree automaton. Since we have established the corresponding automaton notion, we may hope to obtain the corresponding result for feature trees. For the special case of the set of all feature trees, this is the decidability of first-order feature logic. A proof for infinite feature trees can be found in [BS92]. Can the techniques of that proof be combined with the ones of [CD91]?

We add the fact, suggested by one of the referees, that the first-order theory of multitrees is not decidable. This can be shown by employing a proof technique by Ralf Treinen [Tre92].

Appendix

A The Algebra of Multisets

We will reduce \mathcal{J} -recognizability for languages of flat multitrees to a notion of recognizability of finite multisets of pairs. The idea is to identify a flat multitree with a finite multiset of pairs,

$$(A, E) \equiv \{(root, A)\} \sqcup E$$

where *root* is considered like an extra feature. Roughly, the operation of adding edges corresponds to the union operation on multisets.

In all generality, we introduce the algebra $\mathcal{M} = \mathcal{M}(\mathcal{U}_1, \dots, \mathcal{U}_n)$ of finite multisets over n -tuples with components in given sets $\mathcal{U}_1, \dots, \mathcal{U}_n$, for some $n \geq 1$. (Later, we will instantiate $\mathcal{U}_1 = \mathcal{F} \cup \{root\}$ and $\mathcal{U}_2 = \mathcal{S}$.) \mathcal{M} is $n+1$ -sorted, over the the sorts s_1, \dots, s_n and FMS which denote, respectively, the domains $D_{s_1} = \mathcal{U}_1, \dots, D_{s_n} = \mathcal{U}_n$, and $D_{FMS} = \mathbf{N}_{finite}^{\mathcal{U}_1 \times \dots \times \mathcal{U}_n}$ (where \mathbf{N}_{finite}^M denotes the set of finite multisets over M).

The operations of \mathcal{M} are the (associative and commutative) union $\sqcup^{\mathcal{M}}$ of multisets and the creation of a singleton multiset from n elements, one for each component, *i.e.*, $\langle u_1, \dots, u_n \rangle^{\mathcal{M}} = \{(u_1, \dots, u_n)\}$. Thus, they are mappings $\sqcup^{\mathcal{M}} : D_{FMS} \times D_{FMS} \mapsto D_{FMS}$, and $\langle \rangle^{\mathcal{M}} : \mathcal{U}_1 \times \dots \times \mathcal{U}_n \mapsto D_{FMS}$.

Formally, \mathcal{M} is an algebra over the $\{s_1, \dots, s_n, FMS\}$ -sorted signature:

$$\Sigma_{\mathcal{U}_1, \dots, \mathcal{U}_n} = \mathcal{U}_1 \uplus \dots \uplus \mathcal{U}_n \uplus \{\langle \cdot, \dots, \cdot \rangle, \sqcup\}$$

where the constants of sort s_i are just the elements of \mathcal{U}_i , and the two function symbols have the profile: $\sqcup : FMS \times FMS \mapsto FMS$, and $\langle \rangle : s_1 \times \dots \times s_n \mapsto FMS$.

Lemma 3 *The algebra \mathcal{M} is isomorphic to the quotient of the term algebra with the congruence generated by the associativity and commutativity laws for \sqcup ,*

$$\mathcal{M} = T_{\Sigma_{\mathcal{U}_1, \dots, \mathcal{U}_n}} / AC.$$

We define a subset of D_{FMT} of multisets of n -tuples to be **recognizable** if it is recognized by a **finitary** \mathcal{M} -automaton.

It is important to note that the notions of recognizability in $\mathcal{M} = \mathcal{M}(\mathcal{U}_1, \dots, \mathcal{U}_n)$ and $\mathcal{M}(\mathcal{U}_1 \times \dots \times \mathcal{U}_n)$ can be different, namely if $n \geq 2$ and one of the \mathcal{U}_i is infinite.⁹

Now, we consider the special case where $\mathcal{U}_1 = \mathcal{F} \cup \{root\}$ and $\mathcal{U}_2 = \mathcal{S}$, *i.e.*,

$$\mathcal{M} = \mathcal{M}(\mathcal{F} \cup \{root\}, \mathcal{S}).$$

⁹Generally, the finiteness condition for $\mathcal{M}(\mathcal{U}_1 \times \dots \times \mathcal{U}_n)$ -automata is weaker than the one for \mathcal{M} -automata. It may be strictly weaker since cartesian products of finite and co-finite sets need neither be finite nor co-finite. For example, suppose \mathcal{U} to be an infinite set. The cartesian product $\mathcal{U} \times \{1\}$ is neither finite nor co-finite as subset of $\mathcal{U} \times \{0, 1\}$. Thus, the language of the singleton subsets of $\mathcal{U} \times \{1\}$ is not recognizable in the algebra $\mathcal{M}(\mathcal{U} \times \{0, 1\})$, but it is with respect to $\mathcal{M} = \mathcal{M}(\mathcal{U}, \{0, 1\})$.—In fact, it is this finitary-condition which makes the proofs that complicated and non-standard.

Thus, the domains of \mathcal{M} are $D_{s_1}^{\mathcal{M}} = \mathcal{F} \cup \{\text{root}\}$, $D_{s_2}^{\mathcal{M}} = \mathcal{S}$, and $D_{FMS}^{\mathcal{M}} = FMS(\mathcal{F} \cup \{\text{root}\} \times \mathcal{S})$.

We define the injection:

$$I: \mathcal{MT}_1 \rightarrow \mathbf{N}_{finite}(\mathcal{F} \cup \{\text{root}\}) \times \mathcal{S}$$

by $I((A, E)) = \{(root, A)\} \sqcup E$. Thus (writing the operator $\sqcup^{\mathcal{M}}$ infix):

$$I(\Rightarrow^{\mathcal{J}}(t, f, A)) = I(t) \sqcup^{\mathcal{M}} \langle f, A \rangle^{\mathcal{M}}.$$

Lemma 4 (Reduction Lemma) *A language L of flat multitrees is recognizable in \mathcal{J} iff the language $I(L)$ of multisets of pairs is recognizable in \mathcal{M} .*

Proof The difficult direction is from left to right. Given a finitary \mathcal{J} -automaton $(\mathcal{A}, h, Q_{\text{final}})$, where $D_{\mathcal{A}}^{\mathcal{M}T} = Q$ and $D_{\mathcal{A}}^F = P$, we construct a finitary \mathcal{M} -automaton $(\mathcal{A}^*, h^*, Q_{\text{final}})$ such that, for all flat multitrees t :

$$h^*(I(t)) = h(t). \quad (4)$$

This is sufficient to show the recognizability of $I(L)$, since $I(L) = h^{-1}(\mathcal{A}) \cap I(\mathcal{MT}_1)$, and $I(\mathcal{MT}_1)$ is a recognizable set in \mathcal{M} .

We set $D_{s_2}^{\mathcal{A}^*} = Q$, $D_{s_1}^{\mathcal{A}^*} = P \cup \{p_{\text{root}}\}$, and (where *Func* denotes the set of functions generated by the functions $\Rightarrow^{\mathcal{J}}(\cdot, p, q)$; i.e., the smallest set containing these and closed under composition):

$$D_{FMS}^{\mathcal{A}^*} = \text{Func} \uplus Q \uplus \{q_{\perp}\}.$$

The evaluation of \mathcal{A}^* is defined by (we write $\cdot^{\mathcal{A}^*}$ instead of $h^*(\cdot)$ and use the more intuitive infix notation):

$$\begin{aligned} \langle p, q \rangle^{\mathcal{A}^*} &= \Rightarrow^{\mathcal{A}}(\cdot, p, q), \\ \langle p_{\text{root}}, q \rangle^{\mathcal{A}^*} &= q, \\ h_1 \sqcup^{\mathcal{A}^*} h_2 &= h_1 \circ h_2, \\ q \sqcup^{\mathcal{A}^*} h &= h(q), \\ h \sqcup^{\mathcal{A}^*} q &= h(q), \\ q \sqcup^{\mathcal{A}^*} \tilde{q} &= q_{\perp}. \end{aligned}$$

Every function in the interpretations taking q_{\perp} as argument is again mapped to q_{\perp} . Precisely:

$$\begin{aligned} q_{\perp} \sqcup^{\mathcal{A}^*} h &= q_{\perp}, \\ h \sqcup^{\mathcal{A}^*} q_{\perp} &= q_{\perp}, \\ q_{\perp} \sqcup^{\mathcal{A}^*} q &= q_{\perp}, \\ q \sqcup^{\mathcal{A}^*} q_{\perp} &= q_{\perp}, \\ \langle p, q_{\perp} \rangle^{\mathcal{A}^*} &= q_{\perp}, \\ \langle p_{\text{root}}, q_{\perp} \rangle^{\mathcal{A}^*} &= q_{\perp}. \end{aligned}$$

Clearly, \mathcal{A}^* is an AC-automaton, i.e., the operation $\sqcup^{\mathcal{A}^*}$ is associative and commutative. The associativity is trivial for functions as arguments. The commutativity for functions follows

from the *OIT*-axiom, and the associativity for functions by:

$$\begin{aligned} \Rightarrow^{\mathcal{A}} (\cdot, p, q) \sqcup^{\mathcal{A}^*} \Rightarrow^{\mathcal{A}} (\cdot, p_1, q_1) &= \Rightarrow^{\mathcal{A}} (\Rightarrow^{\mathcal{A}} (\cdot, p_1, q_1), p, q) \\ &= \Rightarrow^{\mathcal{A}} (\Rightarrow^{\mathcal{A}} (\cdot, p, q), p_1, q_1) \\ &= \Rightarrow^{\mathcal{A}} (\cdot, p_1, q_1) \sqcup^{\mathcal{A}^*} \Rightarrow^{\mathcal{A}} (\cdot, p, q). \end{aligned}$$

The proof for all possible cases is now easily established.

The identity (4) is now easily verified. Finally, we note that the finitary-condition is preserved from \mathcal{A} to \mathcal{A}^* .

For the other direction, given a finitary \mathcal{M} -automaton \mathcal{A}^* , we will construct a finitary \mathcal{J} -automaton \mathcal{A} satisfying (4). This is sufficient, now, since \mathcal{MT}_1 is a recognizable set in \mathcal{J} . In fact, we will construct an automaton in another algebra.¹⁰ Next, we will introduce this algebra. We resume this proof after having proven Lemma 6.

The algebra \mathcal{J}_{local} of flat multitrees is obtained from the algebra \mathcal{J} by restricting the domain of the third argument from \mathcal{MT} to $\mathcal{S} (\dots = \mathcal{MT}_0)$, and the domain of the first from \mathcal{MT} to \mathcal{MT}_1 , *i.e.*, to flat multitrees instead of arbitrary ones.

That is, the algebra \mathcal{J}_{local} is three-sorted with sorts \mathcal{MT}_1 , F and S . The domains are given by $D_{\mathcal{MT}_1} = \mathcal{MT}_1$, $D_F = \mathcal{F}$, $D_S = \mathcal{S}$. The operation is given by (where E is a finite multiset over pairs in $\mathcal{F} \times \mathcal{S}$):

$$\Rightarrow^{\mathcal{J}_{local}} ((A_1, E), f, A_2) = (A_1, E \sqcup \{(f, A_2)\})$$

(which is equal to $\Rightarrow^{\mathcal{J}} ((A_1, E), f, A_2)$). The signature of \mathcal{J}_{local} is the disjoint union:

$$\Sigma_{local} = \mathcal{S} \uplus \mathcal{F} \uplus \mathcal{S} \uplus \{\Rightarrow\}.$$

Here, the symbols in \mathcal{S} appear twice: they are supposed to be renamed apart. Firstly, they are constants of sort \mathcal{MT}_1 , and secondly, they are constants of sort S . The different functionality is made clear syntactically by writing $A_{\mathcal{MT}_1}$ and A_S , with interpretations $(A_{\mathcal{MT}_1})^{\mathcal{J}_{local}} = (A, \emptyset) \in \mathcal{MT}_0 \subseteq \mathcal{MT}_1$ and $(A_S)^{\mathcal{J}_{local}} = A \in \mathcal{S}$.

The features are constants of sort F and interpreted freely. The profile of the function symbol in \mathcal{J}_{local} is $\Rightarrow: \mathcal{MT}_1 \times F \times \mathcal{S} \rightarrow \mathcal{MT}_1$.

The algebra \mathcal{J}_{local} satisfies the order independence theory (OIT); namely, for all flat multitrees t , features f and symbols A the following holds.

$$\Rightarrow^{\mathcal{J}_{local}} ((\Rightarrow^{\mathcal{J}_{local}} (t, f_1, A_1), f_2, A_2) = \Rightarrow^{\mathcal{J}_{local}} ((\Rightarrow^{\mathcal{J}_{local}} (t, f_2, A_2), f_1, A_1)$$

The following lemma states that we can use the more concrete notion of tree automata.

Lemma 5 *\mathcal{J}_{local} is isomorphic to a quotient term algebra,*

$$\mathcal{J}_{local} = \mathcal{T}_{\Sigma_{local}} / OIT .$$

¹⁰The motivation for the construction of yet another algebra is, roughly, the fact that a symbol $A \in \mathcal{S}$ occurs as a root-labeling as well as a leaf-labeling; these two roles are distinguished in \mathcal{J} -automata, but not in \mathcal{M} -automata.

Again, we define recognizability in \mathcal{J}_{local} in terms of finitary automata.

Lemma 6 *A language of flat multitrees is recognizable in \mathcal{J} iff it is recognizable in \mathcal{J}_{local} .*

Proof We will first modify a finitary \mathcal{J} -automaton \mathcal{A} , where $D_{\mathcal{A}}^{MT} = Q$ and $D_{\mathcal{A}}^F = P$, in order to obtain a finitary \mathcal{J}_{local} -automaton \mathcal{A}^1 such that the two automata (with the same set of final states) will recognize the same languages of flat multitrees. We define the domains of \mathcal{A}^1 by:

$$\begin{aligned} D_S^{\mathcal{A}^1} &= Q, \\ D_{MT_1}^{\mathcal{A}^1} &= Q, \\ D_F^{\mathcal{A}^1} &= P, \end{aligned}$$

and we define the evaluation of \mathcal{A}^1 by (for all $A \in \mathcal{S}, f \in \mathcal{F}$, and for all $q, q' \in Q$ and $p \in P$):

$$\begin{aligned} (A_{MT_1})^{\mathcal{A}^1} &= A^{\mathcal{A}}, \\ (A_S)^{\mathcal{A}^1} &= A^{\mathcal{A}}, \\ f^{\mathcal{A}^1} &= f^{\mathcal{A}}, \\ \Rightarrow^{\mathcal{A}^1} (q, p, q') &= \Rightarrow^{\mathcal{A}} (q, p, q'). \end{aligned}$$

Clearly the finitary-condition and the order independence theory are preserved between \mathcal{A}^1 and \mathcal{A} .

For the other direction, given a finitary \mathcal{J}_{local} -automaton \mathcal{A}^2 (with final states Q_{final}^2 , of sort MT_1), we will define a finitary \mathcal{J}_{local} -automaton \mathcal{A}^1 that recognizes the same language, but has the two properties: $D_{MT_1}^{\mathcal{A}^1} = D_S^{\mathcal{A}^1}$, and, for all symbols A in \mathcal{S} , $(A_{MT_1})^{\mathcal{A}^1} = (A_S)^{\mathcal{A}^1}$. Thanks to these, one can define a \mathcal{J} -automaton \mathcal{A} that accepts the same flat multitrees as \mathcal{A}^1 . Again, this is sufficient since the language MT_1 is recognizable with respect to \mathcal{J} .

We define the domains of \mathcal{A}^1 by:

$$\begin{aligned} D_{MT_1}^{\mathcal{A}^1} &= D_{MT_1}^{\mathcal{A}^2} \times D_S^{\mathcal{A}^2}, \\ D_S^{\mathcal{A}^1} &= D_{MT_1}^{\mathcal{A}^2} \times D_S^{\mathcal{A}^2}, \\ D_F^{\mathcal{A}^1} &= D_F^{\mathcal{A}^2}, \end{aligned}$$

and, after having fixed an arbitrary element $r_{fix} \in D_S^{\mathcal{A}^2}$, we define the evaluation of \mathcal{A}^1 by (for all $A \in \mathcal{S}, f \in \mathcal{F}$, and for all $q, \tilde{q} \in D_{MT_1}^{\mathcal{A}^2}, p \in D_F^{\mathcal{A}^2}$ and $r, \tilde{r} \in D_S^{\mathcal{A}^2}$):

$$\begin{aligned} (A_{MT_1})^{\mathcal{A}^1} &= ((A_{MT_1})^{\mathcal{A}^2}, (A_S)^{\mathcal{A}^2}), \\ (A_S)^{\mathcal{A}^1} &= ((A_{MT_1})^{\mathcal{A}^2}, (A_S)^{\mathcal{A}^2}), \\ f^{\mathcal{A}^1} &= f^{\mathcal{A}^2}, \\ \Rightarrow^{\mathcal{A}^1} ((q, r), p, (\tilde{q}, \tilde{r})) &= (\Rightarrow^{\mathcal{A}^2} (q, p, \tilde{r}), r_{fix}). \end{aligned}$$

As final states of \mathcal{A}^1 we choose:

$$Q_{final}^1 = \{(q, r) \mid q \in Q_{final}^2 \text{ and } r \in D_S^{\mathcal{A}^2}\}.$$

Again, the finiteness condition and the order independence theory are preserved. This concludes the proof of Lemma 6. \square

End of Proof of Reduction Lemma 4

Given a finitary \mathcal{M} -automaton \mathcal{A}^* , we construct a finitary \mathcal{J}_{local} -automaton \mathcal{A} such that $(I(t))^{\mathcal{A}^*} = t^{\mathcal{A}}$ for all flat multitrees t . The domains of \mathcal{A} are: $D_S^{\mathcal{A}} = D_{s_2}^{\mathcal{A}^*}$, $D_F^{\mathcal{A}} = D_{s_1}^{\mathcal{A}^*}$ and $D_{MT_1}^{\mathcal{A}} = D_{FMS}^{\mathcal{A}^*}$.

The evaluation of \mathcal{A} is defined by (where q, p and r are states of \mathcal{A} of sorts MT_1, F and S):

$$\begin{aligned} (A_S)^{\mathcal{A}} &= A^{\mathcal{A}^*}, \\ f^{\mathcal{A}} &= f^{\mathcal{A}^*}, \\ (A_{MT_1})^{\mathcal{A}} &= \langle root^{\mathcal{A}^*}, (A_S)^{\mathcal{A}^*} \rangle^{\mathcal{A}^*}, \\ \Rightarrow^{\mathcal{A}}(q, p, r) &= q \sqcup^{\mathcal{A}^*} \langle p, r \rangle^{\mathcal{A}^*}. \end{aligned}$$

Since \mathcal{A}^* satisfies (AC), \mathcal{A} satisfies (OIT). The finitary-condition is preserved, as well. \square

B Counting in Multisets

Going back to the general framework where $\mathcal{M} = \mathcal{M}(\mathcal{U}_1, \dots, \mathcal{U}_n)$, we will now characterize recognizability in \mathcal{M} , *i.e.*, of languages of finite multisets, by appropriate counting constraints.

We define \mathcal{M} -counting constraints C (written $C(x)$ to indicate that x is the only free variable, which is, logically, a multiset variable) to expressions of the following form:

$$\begin{aligned} C(x) ::= & \text{card} \{ (u_1, \dots, u_n) \in x \mid u_i \in U_i \text{ for all } i \} \in N \\ & \mid C(x) \cap C(x) \\ & \mid C(x) \cup C(x). \end{aligned}$$

Here, N is a recognizable set of natural numbers with respect to the monoid $(\mathbb{N}, +, 0)$, and the sets $U_i \subset \mathcal{U}_i$ are finite or co-finite. The counting constraint

$C(x) \equiv \text{card} \{ (u_1, \dots, u_n) \in x \mid u_i \in U_i \text{ for all } i \} \in N$ holds for the multiset x if the number of tuples (u_1, \dots, u_n) in x such that $u_i \in U_i$ for all $i = 1, \dots, n$ is an element of N . The cardinality operator *card* applies on a multiset of tuples, *i.e.*, counts double occurrences.

The language defined by an \mathcal{M} -counting constraint $C(x)$ is the set of all finite multisets x that satisfy $C(x)$. It is denoted by $L_{\mathcal{M}}(C)$.

Theorem 5 *The family of languages defined by \mathcal{M} -counting constraints is exactly the family of languages of multisets recognizable in \mathcal{M} .*

Proof. Given an \mathcal{M} -counting constraint of the form: $C = \text{card} \{ (u_1, \dots, u_n) \in x \mid u_i \in U_i \text{ for all } i \} \in N$, we will show the recognizability of $L_{\mathcal{M}}(C)$.

We can define a homomorphism $h : \mathcal{M}(\mathcal{U}_1, \dots, \mathcal{U}_n) \rightarrow \mathcal{M}(\{1\}, \dots, \{1\})$ by setting $h(u_i) = \{1\}$ for $u_i \in U_i$, and $h(u_i) = \emptyset$ otherwise.

Furthermore, the homomorphism $J : \mathbb{N}_{finite}^{\{1\}} \times \dots \times \{1\} \rightarrow \mathbb{N}$, given by $J(\{(u_1, \dots, u_n)\}) = 1$ if $(u_1, \dots, u_n) = (1, \dots, 1)$, and $\dots = 0$, otherwise, identifies a multiset consisting of k tuples $(1, \dots, 1)$ with $k \in \mathbb{N}$.

Thus, for all finite multisets of n -tuples $x \in D_{FMT}$,

$$J(h(x)) = \text{card} \{ (u_1, \dots, u_n) \in x \mid u_i \in U_i \text{ for all } i \}.$$

Hence, $L_{\mathcal{M}}(C) = h^{-1}(J^{-1}(N))$. The finitary-condition is invariant under inverse images of homomorphisms. Thus, $L_{\mathcal{M}}(C)$ is recognizable in \mathcal{M} .

For the reverse inclusion, suppose that L is recognized by a finitary \mathcal{M} -automaton $(\mathcal{A}, h, Q_{\text{final}})$ with, say, the set $D_{FMS} = \{q_1, \dots, q_n\}$ of states of sort FMS .

The evaluation of the multiset t by \mathcal{A} leads to the state (written in a notation which is justified by the fact that \mathcal{A} satisfies (AC), even if $\sqcup^{\mathcal{A}}$ is taken over the empty multiset):

$$t^{\mathcal{A}} = \bigsqcup_{(u_1, \dots, u_n) \in t} \mathcal{A} \langle u_1^{\mathcal{A}}, \dots, u_n^{\mathcal{A}} \rangle^{\mathcal{A}}.$$

We define the natural numbers: $a_t(i) = \text{card} \{ (u_1, \dots, u_n) \in t \mid \langle u_1^{\mathcal{A}}, \dots, u_n^{\mathcal{A}} \rangle^{\mathcal{A}} = q_i \}$ and obtain (again thanks to (AC) being satisfied):

$$t^{\mathcal{A}} = \bigsqcup_{i=1}^n \mathcal{A} \bigsqcup_{j=1}^{a_t(i)} q_j.$$

We define a mapping $\nu_t : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ such that $q_{\nu_t(i)} = \bigsqcup_{j=1}^{a_t(i)} q_j$. If $t \in L_{\mathcal{M}}(\mathcal{A})$, then:

$$\bigsqcup_{i=1}^n \mathcal{A} q_{\nu_t(i)} \in Q_{\text{final}}, \quad (5)$$

Generally, for a mapping $\mu : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$, we define, for $i = 1, \dots, n$, the set of natural numbers:

$$N_{\mu}^i = \{m \in \mathbb{N} \mid \bigsqcup_{j=1}^m q_j = q_{\mu(i)}\}.$$

We note that $a_t(i) \in N_{\nu_t}^i$ for $i = 1, \dots, n$. That is, t is an element of the language defined by the \mathcal{M} -counting constraint:

$$\bigwedge_{i=1}^n a_x(i) \in N_{\nu_t}^i.$$

Vice versa, for each mapping μ satisfying the property (5), the language of the \mathcal{M} -counting constraint:

$$\bigwedge_{i=1}^n a_x(i) \in N_{\mu}^i$$

is contained in L . We get $L = L(R)$ where R is the \mathcal{M} -counting constraint:

$$R = \bigvee_{\substack{\mu \\ \text{with (5)}}} \bigwedge_{i=1}^n a_x(i) \in N_{\mu}^i .$$

Since the number of mappings μ with (5) is finite, it only remains to show that the constraints used in R are of the defined form. The constituents $a_i(x)$ are admissible by the finitary-condition of \mathcal{A} . Finally, we have to prove that the sets N_{μ}^i are recognizable with respect to $(\mathbb{N}, +, 0)$. We will construct appropriate automata \mathcal{A}_{μ}^i from \mathcal{A} . We set $D^{\mathcal{A}_{\mu}^i} = \mathcal{Q}$, $0^{\mathcal{A}_{\mu}^i} = \emptyset^{\mathcal{A}}$, $1^{\mathcal{A}_{\mu}^i} = q_i$ and interpret the addition by $\sqcup^{\mathcal{A}}$. As final states we take the singleton $\{q_{\mu(i)}\}$. Then, \mathcal{A}_{μ}^i recognizes N_{μ}^i . \square

Proof of Theorem 2.

For each language L of flat multitrees defined by a counting constraint C we will find an \mathcal{M} -counting constraint C' that defines $I(L)$, and *vice versa*.

Given a counting constraint for flat multitrees of the form:

$$C(x) = \text{card} \{ \varphi \in F \mid \exists y. (x\varphi y \wedge Ty) \} \in N ,$$

we set:

$$C'(x) = \text{card} \{ (\varphi, y) \in x \mid \varphi \in F \wedge y \in T \} \in N \\ \cap \text{card} \{ (\text{root}, y) \in x \mid y \in \mathcal{F} \} = 1 .$$

The case $C = Tx$ is obvious, as well as conjunction and disjunction.

For the other direction, given an \mathcal{M} -counting constraint C' for finite multisets, we will give a constraint C such that $L_{\mathcal{MT}_1}(C_x) = I^{-1}(L_{\mathcal{M}}(C'))$, or, equivalently, $L_{\mathcal{MT}_1}(C) = L_{\mathcal{M}}(C') \cap I(\mathcal{MT}_1)$. We note that the languages of the form $I(L)$ are the multisets containing exactly one pair with first component *root*. Given the \mathcal{M} -counting constraint:

$$C' = \text{card} \{ (\varphi, y) \in x \mid \varphi \in F \wedge y \in T \} \in N ,$$

we have to distinguish the two cases $\text{root} \notin F$ and $\text{root} \in F$. In the first case we set:

$$C = \text{card} \{ \varphi \in F \mid \exists y. (x\varphi y \wedge Ty) \} \in N .$$

In the second case, we note that the set: $N - 1 = \{n - 1 \mid n \in N \text{ and } n \geq 1\}$ is recognizable with respect to $(\mathcal{N}, +, 0)$, and set:

$$C = \text{card} \{ \varphi \in F - \{\text{root}\} \mid \exists y. (x\varphi y \wedge Ty) \} \in N - 1 \\ \cap Tx .$$

In either case C has the required property.

This concludes the proof of Theorem 2, since the reduction lemma (Lemma 4, page 13) and the above theorem (Theorem 5) close the cycle from counting-definable languages L of flat feature trees to those recognizable in \mathcal{J} by feature automata. Namely, according to the above correspondence between counting- and \mathcal{M} -counting constraints, via \mathcal{M} -counting-definable languages $I(L)$, which, according to Theorem 5, are exactly the ones recognizable in \mathcal{M} , back to L according to Lemma 4. \square

References

- [AK86] Hassan Aït-Kaci. An algebraic semantics approach to the effective resolution of type equations. *Theoretical Computer Science*, 45:293–351, 1986.
- [AKP91a] Hassan Aït-Kaci and Andreas Podelski. Functional constraints in LIFE. PRL Research Report 13, Digital Equipment Corporation, Paris Research Laboratory, Rueil-Malmaison, France, 1991.
- [AKP91b] Hassan Aït-Kaci and Andreas Podelski. Towards a meaning of LIFE. In J. Maluszyński and M. Wirsing, editors, *Proceedings of the 3rd International Symposium on Programming Language Implementation and Logic Programming*, Springer LNCS vol. 528, pages 255–274. Springer-Verlag, 1991.
- [AKPS92] Hassan Aït-Kaci, Andreas Podelski, and Gert Smolka. A feature-based constraint system for logic programming with entailment. In *Proceedings of the 5th International Conference on Fifth Generation Computer Systems*, pages 1012–1022, Tokyo, Japan, June 1992. ICOT.
- [BS92] Rolf Backofen and Gert Smolka. A complete and recursive feature theory. Research Report RR-92-30, German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, 6600 Saarbrücken 11, Germany, September 1992.
- [Car92] Bob Carpenter. *The Logic of Typed Feature Structures*, volume 32 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1992.
- [CD91] Hubert Comon and Catherine Delors. Equational formula with membership constraints. Rapport de recherche 648, LRI, Université de Paris Sud, March 1991. To appear in *Information and Computation*.
- [Cou89] Bruno Courcelle. On recognizable sets and tree automata. In Hassan Aït-Kaci-Kaci and Maurice Nivat, editors, *Resolution of Equations in Algebraic Structures, Algebraic Techniques*, volume 1, chapter 3, pages 93–126. Academic Press, 1989.
- [Cou90] Bruno Courcelle. The monadic second-order logic of graphs I: Recognizable sets of finite graphs. *Information and Computation* 85, pages 12-75, 1990.
- [Cou92] Bruno Courcelle. Recognizable sets of unrooted trees. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata, Advances and Open Problems*. Elsevier Science, 1992.
- [Don70] John E. Doner. Tree Acceptors and some of their applications. *Journal of Comp. System Sci.* 4:406-451, 1970.
- [Eil74] Samuel Eilenberg. *Automata, Language and Machine*, volume A of *Applied and Pure Mathematics*. Academic Press, 1974.

- [GS84] Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.
- [Mah88] Michael J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *LICS*, pages 348–457, July 1988.
- [Niv92] Maurice Nivat. Elements of a theory of tree codes. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata, Advances and Open Problems*. Elsevier Science, 1992.
- [Pfe92] Frank Pfenning (editor). *Types in Logic Programming*. MIT Press, 1992.
- [Rou88] William C. Rounds. Set values for unification-based grammar formalisms and logic programming. Report CSLI-88-129, 1988.
- [Smo92] Gert Smolka. Feature constraint logics for unification grammars. *Journal of Logic Programming*, 12:51–87, 1992.
- [ST92] Gert Smolka and Ralf Treinen. Records for logic programming. In *Proceedings of the 1992 Joint International Conference and Symposium on Logic Programming*, Washington, DC, November 1992. The MIT Press, to appear.
- [TW67] John W. Thatcher and John B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, August 1967. Published by Springer-Verlag NY Inc.
- [Trei92] Ralf Treinen. A New Method for Undecidability Proofs of First Order Theories. *Journal of Symbolic Computation*, 14:437–457, 1992.

PRL Research Reports

The following documents may be ordered by regular mail from:

Librarian – Research Reports
Digital Equipment Corporation
Paris Research Laboratory
85, avenue Victor Hugo
92563 Rueil-Malmaison Cedex
France.

It is also possible to obtain them by electronic mail. For more information, send a message whose subject line is `help to doc-server@prl.dec.com` or, from within Digital, to `decprl : : doc-server`.

Research Report 1: *Incremental Computation of Planar Maps*. Michel Gangnet, Jean-Claude Hervé, Thierry Pudet, and Jean-Manuel Van Thong. May 1989.

Research Report 2: *BigNum: A Portable and Efficient Package for Arbitrary-Precision Arithmetic*. Bernard Serpette, Jean Vuillemin, and Jean-Claude Hervé. May 1989.

Research Report 3: *Introduction to Programmable Active Memories*. Patrice Bertin, Didier Roncin, and Jean Vuillemin. June 1989.

Research Report 4: *Compiling Pattern Matching by Term Decomposition*. Laurence Puel and Ascánder Suárez. January 1990.

Research Report 5: *The WAM: A (Real) Tutorial*. Hassan Aït-Kaci. January 1990.[†]

Research Report 6: *Binary Periodic Synchronizing Sequences*. Marcin Skubiszewski. May 1991.

Research Report 7: *The Siphon: Managing Distant Replicated Repositories*. Francis J. Prusker and Edward P. Wobber. May 1991.

Research Report 8: *Constructive Logics. Part I: A Tutorial on Proof Systems and Typed λ -Calculi*. Jean Gallier. May 1991.

Research Report 9: *Constructive Logics. Part II: Linear Logic and Proof Nets*. Jean Gallier. May 1991.

Research Report 10: *Pattern Matching in Order-Sorted Languages*. Delia Kesner. May 1991.

[†]This report is no longer available from PRL. A revised version has now appeared as a book: “Hassan Aït-Kaci, Warren’s Abstract Machine: A Tutorial Reconstruction. MIT Press, Cambridge, MA (1991).”

Research Report 11: *Towards a Meaning of LIFE*. Hassan Aït-Kaci and Andreas Podelski. June 1991 (Revised, October 1992).

Research Report 12: *Residuation and Guarded Rules for Constraint Logic Programming*. Gert Smolka. June 1991.

Research Report 13: *Functions as Passive Constraints in LIFE*. Hassan Aït-Kaci and Andreas Podelski. June 1991 (Revised, November 1992).

Research Report 14: *Automatic Motion Planning for Complex Articulated Bodies*. Jérôme Barraquand. June 1991.

Research Report 15: *A Hardware Implementation of Pure Esterel*. Gérard Berry. July 1991.

Research Report 16: *Contribution à la Résolution Numérique des Équations de Laplace et de la Chaleur*. Jean Vuillemin. February 1992.

Research Report 17: *Inferring Graphical Constraints with Rockit*. Solange Karsenty, James A. Landay, and Chris Weikart. March 1992.

Research Report 18: *Abstract Interpretation by Dynamic Partitioning*. François Bourdoncle. March 1992.

Research Report 19: *Measuring System Performance with Reprogrammable Hardware*. Mark Shand. August 1992.

Research Report 20: *A Feature Constraint System for Logic Programming with Entailment*. Hassan Aït-Kaci, Andreas Podelski, and Gert Smolka. November 1992.

Research Report 21: *The Genericity Theorem and the Notion of Parametricity in the Polymorphic λ -calculus*. Giuseppe Longo, Kathleen Milsted, and Sergei Soloviev. December 1992.

Research Report 22: *Sémantiques des langages impératifs d'ordre supérieur et interprétation abstraite*. François Bourdoncle. January 1993.

Research Report 23: *Dessin à main levée et courbes de Bézier : comparaison des algorithmes de subdivision, modélisation des épaisseurs variables*. Thierry Pudet. January 1993.

Research Report 24: *Programmable Active Memories: a Performance Assessment*. Patrice Bertin, Didier Roncin, and Jean Vuillemin. March 1993.

Research Report 25: *On Circuits and Numbers*. Jean Vuillemin. November 1993.

Research Report 26: *Numerical Valuation of High Dimensional Multivariate European Securities*. Jérôme Barraquand. March 1993.

Research Report 27: *A Database Interface for Complex Objects*. Marcel Holsheimer, Rolf A. de By, and Hassan Aït-Kaci. March 1993.

Research Report 28: *Feature Automata and Sets of Feature Trees*. Joachim Niehren and Andreas Podelski. March 1993.

Research Report 29: *Real Time Fitting of Pressure Brushstrokes*. Thierry Pudet. March 1993.

Research Report 30: *Rollit: An Application Builder*. Solange Karsenty and Chris Weikart. April 1993.

Research Report 31: *Label-Selective λ -Calculus*. Hassan Aït-Kaci and Jacques Garrigue. May 1993.

Research Report 32: *Order-Sorted Feature Theory Unification*. Hassan Aït-Kaci, Andreas Podelski, and Seth Copen Goldstein. May 1993.

Research Report 33: *Path Planning through Variational Dynamic Programming*. Jérôme Barraquand and Pierre Ferbach. September 1993.

Research Report 34: *A penalty function method for constrained motion planning*. Pierre Ferbach and Jérôme Barraquand. September 1993.

Research Report 35: *The Typed Polymorphic Label-Selective λ -Calculus*. Jacques Garrigue and Hassan Aït-Kaci. October 1993.

Research Report 36: *1983–1993: The Wonder Years of Sequential Prolog Implementation*. Peter Van Roy. December 1993.

digital**PARIS RESEARCH LABORATORY**85, Avenue Victor Hugo
92563 RUEIL MALMAISON CEDEX
FRANCE