1

Feature Trees over Arbitrary Structures

Ralf Treinen

ABSTRACT. This paper presents a family of first order feature tree theories, indexed by the theory of the feature labels used to build the trees. A given feature label theory, which is required to carry an appropriate notion of sets, is conservatively extended to a theory of feature trees with the predicates $x[t]y$ (feature t leads from the root of tree x to the tree y, where we require t to be a ground term) and $xt{\downarrow}$ (feature t is defined at the root of tree x; t may be a variable). We present a quantifier elimination procedure to reduce any sentence of the feature tree theory to an equivalent sentence of the feature label theory. Hence, if the feature label theory is decidable, the feature tree theory is too.

If the feature label theory is the theory of infinitely many constants and finite sets over infinitely many constants, we obtain an extension of the feature theory CFT, giving first-class status to arities. As another application, we obtain decidability of the theory of feature trees, where the feature labels are words, and where the language includes the successor function on words, lexical comparison of words and first-class status of arities.

1 Introduction

Feature trees have been introduced as record-like data structures in constraint (logic) programming (Aït-Kaci et al. 1994, Smolka and Treinen 1994), and as models of feature descriptions in computational linguistics (Backofen 1994b, Backofen 1994a). The use of record-like structures in logic programming languages, in the form of so-called ψ-terms (Aït-Kaci 1986), was pioneered by the languages LOGIN (Aït-Kaci and Nasr 1986) and LIFE (Aït-Kaci and Podelski 1991). More recently, Oz (Henz et al. 1995, Smolka 1995) uses a feature constraint system, the semantics of which is directly

Specifying Syntactic Structures
P. Blackburn and M. de Rijke, eds.

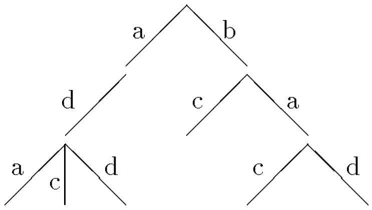

FIGURE 1 A Feature Tree

based on feature trees. In computational linguistics, feature structures have a long history in the field of unification grammars (as described by Smolka 1992).

In both areas, first order predicate logic has been recognized as a powerful description language for feature trees. For the first area, this is clear from the role constraints play in constraint logic programming (Jaffar and Lassez 1987) and in concurrent constrained-based languages (Smolka 1995), and, although in the second area various approaches have been proposed, Smolka and Aït-Kaci 1989, Johnson 1988 and Smolka 1992 have all advocated the use of predicate logic as feature description languages. Moreover, Backofen 1994a argues that predicate logic is the right language to express phenomena in both fields, and that feature trees constitute the canonical semantical model.

Feature trees (Aït-Kaci et al. 1994) are possibly infinite, finitely branching trees, where the edges carry labels taken from some given set of feature symbols. Features are functional, that is all edges departing from the same node have different labels. In contrast to the usual definition from the literature, we will omit in this paper the labeling of nodes by so-called sort symbols.

Different first order languages for feature trees have been studied. The most basic class of predicate symbols, which is contained in any first order feature language, consists of binary relation symbols $x[f]y$ for every feature symbol f. In the standard model of feature trees, the denotation of this predicate is "*y is the direct subtree of x under edge f*". The feature theory *FT* (Aït-Kaci et al. 1994) is an axiomatization of feature trees based on this language together with equality and sort predicates. The feature theory *CFT* (Smolka and Treinen 1994) uses a much more expressive language which extends *FT* by so-called arity constraints $x\{f_1, \ldots, f_n\}$. The denotation of such a constraint in the standard model is "*x has exactly edges labeled by $f_1, \ldots, f_n$ departing from its root*". Furthermore, regular path expressions (which contain an implicit existential quantification over feature paths, Backofen 1994b), and subsumption ordering constraints (Dörre and Rounds 1992, Dörre 1994) have been considered. Finally, the language F (Treinen 1993) contains a ternary feature predicate $x[y]z$. Using quantification over features, all other feature theories can be embedded in the theory F (Treinen 1993, Backofen 1994a).

With the establishment of first order logic as feature description language, various concrete problems concerning logical theories of feature trees were attacked. Much of this works treats the decision problems for various syntactically characterized fragments of the theory of feature trees. Satisfiability of existentially quantified conjunctions of atomic constraints (so-called basic constraints) and entailment between basic constraints is efficiently decidable for the languages *FT* (Aït-Kaci et al. 1994) and *CFT* (Smolka and Treinen 1994), and satisfiability of regular path constraints (Backofen 1994b) and weak subsumption constraints (Dörre 1994) is decidable, while it is undecidable for subsumption constraints (Dörre 1993). Such results lead to a more general question: whether or not the full first order theories of these languages is decidable. An affirmative answer was given for the case of *FT* (Backofen and Smolka 1995) and *CFT* (Backofen and Treinen 1994, Backofen 1995). Not surprisingly, however, the full first order theory of feature trees over F is undecidable, although the existential fragment of the theory is NP-complete, even with arity constraints as additional primitive notions (Treinen 1993).

The reason for the undecidability of F is the fact that it allows one to quantify over the direct subtrees of a tree. Taking $x \prec y$ ("*x is a direct subtree of y*") as abbreviation of $\exists f\, y[f]x$, we can define for trees x with only finitely many different subtrees (rational trees) the predicate "*x is a subtree of y*" by

$$\forall z \left(y \prec z \wedge \forall y_1, y_2\, (y_1 \prec y_2 \prec z \rightarrow y_1 \prec z) \rightarrow x \prec z \right)$$

Here, the idea is to "abuse" feature trees as sets, taking the direct subtrees of a tree as the elements of a set. Note that z fulfills the hypothesis in the above formula exactly if the "set" z contains y and is transitive, and that hence the transitive closure of y is the smallest z which satisfies the hypothesis.

Thus, we can easily show (for example, with the method of Treinen 1992) the undecidability of the theory of feature trees in the language of F. Consequently, in order to get a decidable sub-theory of F, we have to restrict the use of quantification over features.

The first contribution of this paper is the formulation of a decidable theory of feature trees which lies between *CFT* and F. The idea is to allow quantification over features only in order to state which features are defined, but not to quantify over the direct subtrees of a tree. More precisely, we will define the *restricted theory* of feature trees as the set of formulae where t

in $x[t]y$ is always a ground term, but where atomic constraints $xf\downarrow$ ("f *is defined on* x"), where f may be a variable, are still allowed. This situation is similar to that in Process Logic, where unrestricted quantification over path and state variables lead immediately to an undecidable validity problem, while a syntactic restriction leads to a decidable sub-logic (Parikh 1978).

In spite of these restrictions, the resulting theory is an essential extension of *CFT*. As a first example, note that we can encode a *CFT* arity constraint $x\{f_1, \ldots, f_n\}$ as $\forall f\,(xf\downarrow \leftrightarrow \bigvee_{i=1}^{n} f \doteq f_i)$. Once this is observed, we can move beyond the expressivity of *CFT*. We can make statements about the arities of trees; for example, we can say that the arity of x is contained in the arity of y by

$$\forall f\ (xf\downarrow \rightarrow yf\downarrow)$$

As a second example, the following formula expresses that x has exactly three features:

$$\exists f_1, f_2, f_3 \Big(f_1 \not\doteq f_2 \wedge f_2 \not\doteq f_3 \wedge f_1 \not\doteq f_3 \wedge \forall g (xg\downarrow \leftrightarrow [g \doteq f_1 \vee g \doteq f_2 \vee g \doteq f_3])\Big)$$

From these examples, one gets the idea that the theory of sets of feature symbols is hidden in our restricted theory of feature trees. This leads to the second contribution of our approach, which we now explain in three steps.

The first step is to realize that, in order to decide the validity of first order sentences over feature trees, we can save some work if we employ an existing decision algorithm for the theory of finite sets over infinitely many constants. Since this theory is easily encoded in the theory WS1S, the weak second order monadic theory of one successor function, the existence of such an algorithm follows immediately from Büchi's result on the decidability of WS1S (Büchi 1960)[1].

The following examples give an idea why logical statements involving feature trees can be reduced to logical statements on sets of features. Let x, y, z denote variables ranging over feature trees, f, g, h range over features and F, G, H range over sets of features. First, the formula

$$\exists x, y \Big(\forall f\ (xf\downarrow \rightarrow yf\downarrow) \wedge \neg\forall g\ (yg\downarrow \rightarrow xg\downarrow)\Big)$$

does not involve any tree construction. This formula is just about the sets of features defined at the roots of x and y, and hence can be translated to:

$$\exists F, G \Big(\forall f\ (f \mathrel{\dot\in} F \rightarrow f \mathrel{\dot\in} G) \wedge \neg\forall g\ (g \mathrel{\dot\in} G \rightarrow g \mathrel{\dot\in} F)\Big)$$

[1] The reader shouldn't be confused by the fact that we are apparently mixing first and second order structures. A second order structure can always be considered as a two-sorted first order structure, with one sort for the elements, and another sort for the sets. Only in the context of *classes of* structures does it really make sense to distinguish first order from second order structures.

Formulae like the above subformula ($\forall f\ (xf{\downarrow}\ldots)$, where the feature tree x is only used as a set of features, will be called primitive formulae.

The formula

$$\exists x\ (x[a]x \wedge x[b]y \wedge \neg xh{\downarrow}) \tag{1}$$

where a and b are two different constants, is clearly satisfiable if we can find a set which contains a and b, but not h. Hence, (1) can be reduced to

$$\exists F\ (a \mathrel{\dot{\in}} F \wedge b \mathrel{\dot{\in}} F \wedge \neg h \mathrel{\dot{\in}} F) \tag{2}$$

In the setting we have defined so far, this is equivalent to $a \not\doteq h \wedge b \not\doteq h$.

The next step is to generalize this idea to the situation where we have *some* structure of feature symbols and finite sets of feature symbols given, and to build the feature trees with the feature labels we find in the given feature label structure. Hence, we now obtain a family of feature tree structures, indexed by the feature label structures. This is a well-known situation, for instance in constraint domains for programming languages (Smolka 1995), where feature constraints are not isolated but come in combination with other constraint domains like numbers and words.

Hence, our decision procedure now decides the validity of a sentence of the feature tree theory *relative* to the theory of the feature labels. As a consequence, our feature tree theory is decidable if the feature label theory is. Relatively little needs to be done to adapt the reduction procedure to this more general case. The only problem is that two different ground terms, like the constants a and b in example (1) above, need not necessarily denote semantically different elements. This means that we have to consider the two cases $a \doteq b$ and $a \not\doteq b$. In the first case, $a \doteq b$ and the functionality of features yield $x \doteq y$. Hence, we can eliminate x, and obtain for the first case

$$a \doteq b \wedge y[a]y \wedge y[b]y \wedge \neg yh{\downarrow}$$

In the second case, we get the same reduction as before:

$$a \not\doteq b \wedge \exists F\ (a \mathrel{\dot{\in}} F \wedge b \mathrel{\dot{\in}} F \wedge \neg h \mathrel{\dot{\in}} F)$$

The feature label structure may be equipped with operations and predicate symbols of their own, which of course can be used in the feature tree structure as well. We could for instance take as feature label structure WS2S, that is the structure of words over the alphabet $\{a, b\}$, finite sets of words, and successor functions for every symbol of the alphabet. Since the membership predicate in any regular language is definable in the theory of this structure, we can express in this feature tree theory for any regular language L that the arity of some x is contained in L.

So far, feature trees have been finitely branching trees; that is, we took as possible arities all finite sets of features. The third step is to generalize this to an arbitrary notion of arities. That is, we assume that the feature label structure comes with a notion of sets, where we only require that there are at least two different sets. From this, we construct the feature trees such that the arities of the trees are always sets of the given feature label structure. For instance, we get as before the *finitely branching* feature trees if the feature label structure contains all the *finite* sets of feature trees. If we take as feature label structure natural numbers and all the initial segments of the natural numbers, that is sets of the form $\{1, \ldots, n\}$, we get a structure of feature trees where at every node the edges are consecutively numbered. In example (1) above, this has the consequence that we cannot reduce (2) to $a \dot{\neq} h \wedge b \dot{\neq} h$. Instead, the satisfiability of (3) depends on the theory of the feature label structure.

As another example, consider

$$\exists x, y\, (x[f]x \wedge y[f]y \wedge x \dot{\neq} y) \tag{3}$$

Here, we will make a case distinction: Either both x and y have the arity $\{f\}$, that is f is the only feature defined, or at least one of them has a greater arity. In the first case both variables are called tight, in the second case a variable with arity greater than $\{f\}$ is called sloppy. Intuitively, a sloppy variable has features for which there are no constraints. For the case that both variables are tight, the formula can not be satisfied. This is a consequence of the fact that the formula $x[f]x \wedge arity(x, \{f\})$, a so-called *determinant* (Smolka and Treinen 1994), has a *unique* solution. In the other case, the formula is clearly satisfied, since we can use the unconstrained features of x, respectively y, to make both values different. Hence, we can translate (3) to the formula which states that this other case is indeed possible:

$$\exists F, g\, (f \dot{\in} F \wedge \neg g \dot{\in} F) \tag{4}$$

Up to now, we have been talking about the feature tree structures defined upon some feature label structure. The quantifier elimination procedure we are going to present will be based on an axiomatization *FX* only; no other properties of the structures will be used for the justification of the procedure. The axiomatization is not subject to the syntactic restriction we imposed on the input formulae to the procedure, that is the axioms may contain subformulae $x[t]y$ where t is non-ground.

The quantifier elimination procedure proposed in this paper takes another road than the quantifier eliminations which have been given for the feature theories *FT* (Backofen and Smolka 1995) and *CFT* (Backofen 1995). We believe that, in the case of *FT* and *CFT*, our procedure is simpler than the existing ones for these theories. The difference lies in the way the procedure deals with the fact that these feature theories themselves do not have the property of quantifier elimination. A theory T is defined to have the *property of quantifier elimination* (Hodges 1993), if for every variable x and atomic formulae $\phi_1, \ldots, \phi_n$ there is a quantifier-free formula ψ such that $T \models \exists x\, (\phi_1 \wedge \ldots \wedge \phi_n) \leftrightarrow \psi$. An effective procedure to compute this ψ

yields immediately a decision procedure for T, provided ψ does not contain new free variables, and provided *True* and *False* are the only quantifier-free formulae. A simple counterexample, showing that for instance FT does not have the property of quantifier elimination, is

$$\exists x\ (y[l]x \wedge xk{\downarrow}) \tag{5}$$

We can not simply eliminate x, since we need it to express a property of the free variable y, which we must not drop.

The classical way to solve this problem is to extend the language, such that non-reducible formulae like (5) become atomic formulae in the extended language. In our example, this means that we have to add so-called *path-constraints* like $y(lk){\downarrow}$ to the language. This solution was chosen by Backofen and Smolka 1995 and Backofen 1995.

We will use another idea: we exploit the functionality of features to trade an existential quantifier, in the above situation, for a universal quantifier, and transform (5) into:

$$yl{\downarrow} \wedge \forall x\ (y[l]x \rightarrow xk{\downarrow})$$

We can benefit from this quantifier-switching if we consider the elimination of *blocks* of quantifiers of the same kind. This idea has already been used, for instance by Malc'ev 1971 and Comon and Lescanne 1989. We will consider formulae in prenex normal form, for instance

$$\exists \cdots \exists \forall \cdots \forall \exists \cdots \exists \phi$$

where ϕ is quantifier-free. If we can transform $\exists \cdots \exists \phi$ into a formula of the form $\forall \cdots \forall \psi$ for some quantifier-free ψ, then we have reduced the number of *quantifier alternations* from 2 to 1, although the total number of quantifiers might have increased.

The rest of the paper is organized as follows: Section 2 fixes the necessary notions from predicate logic. In Section 3, we define by an axiom the class of feature label structures, which will be called *admissible parameter structures* in the rest of the paper. In Section 4 we construct the standard model of feature trees over some arbitrary admissible parameter structure, present the axiomatization FX and show that the feature tree structure is a model of FX. Some basic properties of the axiomatization FX are stated in Section 5. The overall structure of the quantifier elimination procedure is presented in Section 6, the details are given in Section 7.

2 Preliminaries

We consider many-sorted predicate logic with equality. We use a number of standard shortcuts from predicate logic. We write $\tilde{\forall}\, \phi$ for the universal closure of ϕ. We write $\exists \bar{x}\ \phi$, where $\bar{x} = (x_1, \ldots, x_n)$ is a list of variables, for $\exists x_1\ \ldots \exists x_n\ \phi$ ($\forall \bar{x}\ \phi$ is defined analogously). We sometimes use the notation $\exists X\, \phi$, where X is a finite set of variables, for $\exists \bar{x}\, \phi$, where $\bar{x}$ is some linear arrangement of X. Instead of writing the sort with every quantified variable, as in "$\forall x \in S \ldots$", we will introduce naming conventions which allow us to directly read off the sort of a variable. As usual, variables may be decorated with sub-and superscripts. Lists of variables will be denoted with an overstrike as in $\bar{x}$.

The junctors $\wedge, \vee$ take precedence over (bind tighter than) $\leftrightarrow, \rightarrow$. Negation $\neg$ and quantors bind tightest. It is understood that conjunction is commutative and associative. Consequently, we identify a conjunction of formulae with the multiset of its conjuncts. We use notions like $\psi \in \phi$ or $\psi \subseteq \phi$, where ϕ is a conjunction, accordingly.

We write the negation of $x \doteq y$ as $x \not\doteq y$. We consider equality as symmetrical, that is we identify $x \doteq y$ with $y \doteq x$ (and hence, $x \not\doteq y$ with $y \not\doteq x$). The reader should be aware, that $x \doteq y$ and $x \not\doteq y$ are formulae of our object logic, while $x = y$, respectively $x \neq y$, is a mathematical statement, expressing that the two variables x, y are syntactically identical, respectively, distinct.

We write $\mathtt{fr}(\phi)$ for the set of free variables of ϕ, and $\phi[y/x]$ denotes the formula that is obtained from ϕ by replacing every occurrence of x by y, after possibly renaming bound variables to avoid capture.

An assignment α is a X-update of an assignment α', where X is a set of variables, if $\alpha(x) = \alpha'(x)$ for all variables $x \notin X$. We write $\alpha[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]$ for the $\{x_1, \ldots, x_n\}$-update of α which assigns a_i to x_i, respectively.

3 Admissible Parameter Structures

In this section, we specify the class of parameter structures which we want to allow as a basis for the construction of feature trees.

Definition 3.1 *[Admissible parameter signature] The signature $\Sigma = \langle S_\Sigma, F_\Sigma, R_\Sigma \rangle$ is an* admissible parameter signature*, if S_Σ contains at least the two sorts Feat and Set, and R_Σ contains at least the relational symbol $\mathsf{Feat} \mathrel{\dot{\in}} \mathsf{Set}$, that is the binary infix relation symbol $\dot{\in}$ of profile $\mathsf{Feat}, \mathsf{Set}$.*

The sort Feat is intended to denote the features, and the sort Set is intended to denote the sets of features. In this sense, $\dot{\in}$ can be thought of as the usual elementhood relation.

Small letters from the middle of the alphabet $f, g, h, \ldots$ are variables of sort Feat, and capital letters from the middle of the alphabet $F, G, H, \ldots$ are variables of sort Set.

The only requirement on the class of admissible parameter structures is that they contain at least two (observationally) different sets:

(S2)	$\exists F, G, f\,(f \mathrel{\dot{\in}} F \wedge \neg f \mathrel{\dot{\in}} G)$

Definition 3.2 *[Admissible Parameter Structure] Let Σ be an admissible parameter signature. We call a Σ-structure $\mathcal{B}$ an* admissible parameter structure, *if $\mathcal{B} \models (S2)$.*

This is in two respects weaker than what is usually required by axioms systems of second order logic (Andrews 1986). First, we don't require extensionality; that is, two different sets may have the same elements. Second, axiom (S2) is much weaker than the usual comprehension axiom of second order logic which states that every formula denotes a set. Note that, as a consequence of (S2), every admissible parameter structure contains at least one element of sort Feat.

Examples of admissible parameter signatures and algebras are

1. The signature Σ_C consists of an infinite set C of Feat-constants and the $\in$ predicate symbol. The algebra $\mathcal{B}_C$ assigns C to Feat, every constant of C to itself, the powerset over C to Set, and the elementship relation to $\in$.
2. Σ_F and $\mathcal{B}_F$ are defined as above with the only difference that Set is interpreted as the class of *finite* sets over C.
3. The signature Σ_N contains the constant 1 of sort Feat, the unary function symbol $succ$ of profile $\mathsf{Feat} \rightarrow \mathsf{Feat}$, and $\in$. The algebra $\mathcal{B}_N$ assigns the set of natural numbers to Feat, the number 1 to the constant 1 and the successor function to $succ$. Set denotes the class of initial segments of natural numbers (that is, sets of the form $\{1, \ldots, n\}$), and $\in$ denotes elementship.
4. The signature Σ_S contains the constant ϵ of sort Feat, finitely many function symbols $succ_i$, $1 \leq i \leq n$, of profile $\mathsf{Feat} \rightarrow \mathsf{Feat}$, two predicate symbols $\leq_{pre}$ and $\leq_{lex}$, and $\in$. The algebra $\mathcal{B}_S$ assigns the set $\{1, \ldots, n\}^*$ to Feat, the empty word to ϵ, the function $\lambda x.xn$ to $succ_n$, the prefix (respectively lexical) ordering to $\leq_{pre}$, respectively $\leq_{lex}$, the powerset of $\{1, \ldots, n\}^*$ to Set, and elementship to $\in$.

4 Feature Tree Structures

In this section we give the definition of a standard model of features trees over some given admissible parameter structure. We also present a set of axioms for feature trees, and prove that the standard model of feature trees is indeed a model of this axiomatization. No other properties of the feature tree model than the axiomatization will be used for the justification of the quantifier elimination procedure to be presented in the next sections.

Definition 4.1 *[Tree signature] For a given admissible parameter signature Σ, we define the* tree signature *$\Sigma^\dagger = \langle S_{\Sigma^\dagger}, F_{\Sigma^\dagger}, R_{\Sigma^\dagger}\rangle$ by*

$$S_{\Sigma^\dagger} \quad = \quad S_\Sigma \overset{+}{\cup} \{\mathit{Tree}\}$$

$$F_{\Sigma^\dagger} = F_\Sigma$$
$$R_{\Sigma^\dagger} = R_\Sigma \overset{+}{\cup} \{\mathit{Tree}[\mathit{Feat}]\mathit{Tree}, \mathit{Tree}\,\mathit{Feat}{\downarrow}\}$$

In the standard model to be defined below, the sort symbol Tree denotes a set of trees. Small letters at the end of the alphabet $(x, y, z \ldots)$ denote Tree-variables. Note that the only Tree-terms are the Tree-variables, and that any $\Sigma^\dagger$-formula without Tree-variables is in fact a Σ-formula. We write the negation of $xt{\downarrow}$ as $xt{\uparrow}$.

Definition 4.2 *[Tree] For a set M, a set $\tau \subseteq M^*$ of finite M-words is called a* tree over M *if it is prefix-closed, that is if $vw \in \tau$ implies $v \in \tau$ for all $v, w \in M^*$. $\mathcal{T}(M)$ denotes the set of trees over M.*

Note that every tree contains the empty word ϵ and hence is non-empty, and that a tree may be infinite. This is of course the usual definition of trees—the tree in Figure 1, for instance, is

$$\{\epsilon, a, b, ad, bc, ba, ada, adc, add, bac, bad\}$$

Definition 4.3 *[Admissible Tree] For an admissible parameter structure $\mathcal{B}$, an* admissible tree over $\mathsf{Feat}^{\mathcal{B}}$ *is a tree $\tau \in \mathcal{T}(\mathit{Feat}^{\mathcal{B}})$, such that*

$$\textit{for all } v \in \tau \textit{ exists } M \in \mathit{Set}^{\mathcal{B}} \textit{ with:} \quad \beta \,\dot{\in}^{\mathcal{B}} M \Leftrightarrow v\beta \in \tau$$

$\mathcal{AT}(\mathit{Feat}^{\mathcal{B}})$ denotes the set of admissible trees over $\mathit{Feat}^{\mathcal{B}}$.

Intuitively, this means that the set of features defined at some node of an admissible tree must be licensed by the denotation of Set in the admissible parameter structure $\mathcal{B}$. If we take, for example, an admissible structure $\mathcal{B}$ where $\mathsf{Set}^{\mathcal{B}}$ is the class of finite subsets of $\mathsf{Feat}^{\mathcal{B}}$, then $\mathcal{AT}(\mathsf{Feat}^{\mathcal{B}})$ contains exactly the finitely branching trees over $\mathsf{Feat}^{\mathcal{B}}$.

Definition 4.4 *[Feature tree structure] For any admissible Σ-structure B, we define the $\Sigma^\dagger$-structure $B^\dagger$ by*

1. $\mathcal{B}^\dagger|_\Sigma = \mathcal{B}$,
2. $\mathit{Tree}^{\mathcal{B}^\dagger} = \mathcal{AT}(\mathit{Feat}^{\mathcal{B}})$,
3. $\tau[\beta]^{\mathcal{B}^\dagger}\sigma$ *iff* $\sigma = \{v \mid \beta v \in \tau\}$,
4. $\tau\beta\downarrow^{\mathcal{B}^\dagger}$ *iff* $\beta \in \tau$.

Hence, $\mathcal{B}^\dagger$ is a conservative extension of $\mathcal{B}$.

The first axiom gives an explicit definition for the $\cdot\cdot\!\downarrow$ predicate:

$(\downarrow)$	$\forall x, f\ (xf{\downarrow} \leftrightarrow \exists y\ x[f]y)$

The next axiom scheme expresses that every feature is functional:

(F)	$\forall x, y, z\ (x[t]y \wedge x[t]z \rightarrow y \doteq z)$	where t is ground.

Syntactic Convention $arity(x, F) := \forall f\ (xf{\downarrow} \leftrightarrow f \mathrel{\dot\in} F)$

If $\bar{x} = (x_1, \ldots, x_n)$ and $\bar{F} = (F_1, \ldots, F_n)$, we write $arity(\bar{x}, \bar{F})$ for $\bigwedge_{i=1}^{n} arity(x_i, F_i)$.

The next axiom states that every tree has an arity, and hence reflects the fact that we consider admissible trees only.

(A)	$\forall x\, \exists F\ arity(x, F)$

By construction, we get immediately:

Proposition 4.5 *For any admissible Σ-structure $\mathcal{B}$, we have that $\mathcal{B}^\dagger \models$ $(\downarrow), (F), (A)$.*

The next axiom scheme asserts that certain formulae indeed have a solution in the domain of feature trees.

Definition 4.6 *[Graph, Constrained variable] A conjunction γ of formulae of the form $x[t]y$ is a called a* graph. *For a graph γ, let $\mathsf{co}(\gamma) := \{x \mid x[t]y \in \gamma$ for some t and $y\}$ be the set of variables* constrained by *γ.*

Syntactic Convention For a graph γ and variable x, we define

$$\begin{array}{lcl} F_\gamma^x & := & \{t \mid x[t]y \in \gamma \text{ for some variable } y\} \\ \Delta_\gamma & := & \{t \mathrel{\dot{\neq}} s \mid \text{ there are } x[t]y, x[s]z \in \gamma \text{ where } t \neq s \text{ or } y \neq z\} \end{array}$$

For instance,

$$\gamma := x[a(f)]y \wedge x[b(g, a(f))]z \wedge y[a(a(f))]x \wedge y[a(a(f))]y$$

is a graph with $\mathsf{co}(\gamma) = \{x, y\}$, $F_\gamma^x = \{a(f), b(g, a(f))\}$, $F_\gamma^y = \{a(a(f))\}$, $F_\gamma^z = \emptyset$, and $\Delta_\gamma = a(f) \mathrel{\dot{\neq}} b(g, a(f))$.

(E)	$\tilde{\forall} \left[\left(\Delta_\gamma \wedge \bigwedge_{i=1}^{n} \bigwedge_{a \in F_\gamma^{x_i}} a \mathrel{\dot\in} F_i \right) \rightarrow \exists \mathsf{co}(\gamma) \left(\gamma \wedge \bigwedge_{i=1}^{n} arity(x_i, F_i) \right) \right]$
	where γ is a graph with $\mathsf{co}(\gamma) = \{x_1, \ldots, x_n\}$.

An example of axiom scheme (E) is

$$\begin{array}{lr} (6) & \forall z, f_1, f_2, g, F, G\, (f_1 \mathrel{\dot{\neq}} f_2 \wedge f_1 \mathrel{\dot\in} F \wedge f_2 \mathrel{\dot\in} F \wedge g \mathrel{\dot\in} G \\ & \rightarrow \exists x_1, x_2\, (x_1[f_1]x_2 \wedge x_1[f_2]z \wedge x_2[g]x_1 \wedge \\ & arity(x_1, F) \wedge arity(x_2, G))) \end{array}$$

Proposition 4.7 *$\mathcal{T}(M)$ with the subset relation is a cpo.*

(see, for example, Gunter and Scott 1990 for definition and basic properties of cpos). Note, that in general $\mathcal{AT}(M)$ does not constitute a sub-cpo of

$\mathcal{T}(M)$. Obviously, the set of compact elements of $\mathcal{T}(M)$ are exactly the finite sets in $\mathcal{T}(M)$, and $\mathcal{T}(M)$ is an algebraic cpo.

Lemma 4.8 *For any admissible Σ-structure $\mathcal{B}$, we have $\mathcal{B}^\dagger \models (E)$.*

Proof: (Sketch) Let γ be a graph, $\mathsf{co}(\gamma) = \{x_1, \ldots, x_n\}$, and $\mathcal{B}^\dagger, \alpha \models \Delta_\gamma \wedge \bigwedge_{i=1}^n \bigwedge_{a \in F_\gamma^{x_i}} a \mathrel{\dot\in} F_i$. We construct $\tau_1, \ldots, \tau_n \in \mathcal{AT}(\mathsf{Feat}^{\mathcal{B}})$, such that

$$\mathcal{B}^\dagger, \alpha[x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n] \models \gamma \wedge \bigwedge_{i=1}^n \wedge arity(x_i, F_i) \tag{7}$$

We define the operator Φ: $(\mathcal{T}(\mathsf{Feat}^{\mathcal{B}}))^n \rightarrow (\mathcal{T}(\mathsf{Feat}^{\mathcal{B}}))^n$ by its n components $pr_i \circ \Phi$. For given i, let $\{x_i[t_1]z_1, \ldots, x_i[t_m]z_m\}$ be the set of atoms in γ which constrain x_i.

$$pr_i \circ \Phi(\nu_1, \ldots, \nu_n) = \{\epsilon\} \cup \beta_1\sigma_1 \cup \ldots \cup \beta_m\sigma_m \cup \{\beta \in \mathsf{Feat}^{\mathcal{B}} \mid \beta \mathrel{\dot\in}^{\mathcal{B}} \alpha(F_i)\}$$

where β_j is the evaluation of t_j in $\mathcal{B}, \alpha$, and where we define $\sigma_j := \nu_k$ if $z_j = x_k$ for some $1 \leq k \leq n$, and otherwise $\sigma_j := \alpha(z_j)$. As usual $\beta_j\sigma_j$ is an abbreviation for $\{\beta_j v \mid v \in \sigma_j\}$.

Φ is obviously continuous, hence we can define $(\tau_1, \ldots, \tau_n)$ as the least fixed point of Φ. By construction, $\tau_i \in \mathcal{AT}(\mathsf{Feat}^{\mathcal{B}})$ for all i. Since $\mathcal{B}^\dagger, \alpha \models \Delta_\gamma$, all β_j for given i are different. Hence, (7) holds. □

As an example of this construction, consider the formula (6). Let $\alpha(z) = \{\epsilon, e, ee\}$, $\alpha(f_1) = a$, $\alpha(f_2) = b$, $\alpha(g) = c$, $\alpha(F) = \{a, b\}$ and $\alpha(G) = \{c, d\}$. In this case, we define Φ by

$$\begin{aligned}
pr_1 \circ \Phi(\nu_1, \nu_2) &= \{\epsilon\} \cup a\nu_2 \cup b\{\epsilon, e, ee\} \cup \{a, b\} \\
&= \{\epsilon, b, be, bee\} \cup a\nu_2 \\
pr_2 \circ \Phi(\nu_1, \nu_2) &= \{\epsilon\} \cup c\nu_1 \cup \{c, d\} \\
&= \{\epsilon, d\} \cup c\nu_1
\end{aligned}$$

The least fixed point of Φ is (L_1, L_2), where L_1 is the prefix-closure of $(ac)^*(bee \cup ad)$, and L_2 is the prefix-closure of $(ca)^*(d \cup cbee)$.

Syntactic Convention Let M be a finite set of Feat-terms.

$$arity(x, M) := \forall f\ (xf{\downarrow} \leftrightarrow \bigvee_{a \in M} f \doteq a)$$

As above, this notion generalizes to $arity(\bar{x}, \bar{M})$.

Definition 4.9 *A* determinant δ *is a formula*

$$\gamma \wedge \bigwedge_{x \in \mathsf{co}(\gamma)} arity(x, F_\gamma^x)$$

where γ is a graph and has only free variables of sort Tree.

In other words, for every constraint $x[t]y$ in a determinant, the term t must be ground.

For instance, from the following three formulae

$$x[a(c)]y \wedge x[d]z \wedge y[a(d))]x \wedge arity(x, \{a(c), d\}) \wedge arity(y, \{a(d)\})$$
$$x[a(f)]y \wedge arity(x, \{a(f)\})$$
$$x[a(c)]y \wedge x[b(d, a(c))]z \wedge arity(x, \{a(c)\})$$

only the first one is a determinant (since f denotes a variable).

The last axiom scheme expresses that determinants have at most one solution in the constrained variables.

Syntactic Convention $\exists^{\leq 1}\bar{x}\ \phi$ is an abbreviation for

$$\forall \bar{x}, \bar{y}\ (\phi(\bar{x}) \wedge \phi(\bar{y}) \rightarrow \bar{x} \doteq \bar{y})$$

where $\bar{y}$ is some list of distinct variables as long as $\bar{x}$, and disjoint to $\mathtt{fr}(\phi)$.

$\exists^{\leq 1}\bar{x}\ \phi$ reads "there is at most one tuple $\bar{x}$, such that ϕ".

(U)	$\tilde{\forall}\ (\Delta_\gamma \rightarrow \exists^{\leq 1}\mathtt{co}(\delta)\ \delta)$	where δ is a determinant.

An example of (U) is

$$\begin{aligned}\forall z\ (a_1 \not\doteq a_2 \rightarrow \exists^{\leq 1} x, y\ (&x[a_1]y \wedge x[a_2]z \wedge y[b]x \\ &\wedge arity(x, \{a_1, a_2\}) \wedge arity(y, \{b\})))\end{aligned}$$

Note, that (U) does not state that a determinant always has a solution. In the above example, it might be the case that, for example, the "set" $\{b\}$ does not exist, that is that $\exists F\ \forall x\ (x \in F \leftrightarrow x \doteq b)$ does not hold in the parameter structure. In this case, the determinant does not have a solution due to axiom (A).

Lemma 4.10 *For any admissible Σ-structure $\mathcal{B}$, we have $\mathcal{B}^\dagger \models (U)$.*

Proof: (Sketch) We split the determinant δ into $\gamma \wedge \rho$, where γ is a graph and ρ is a conjunction of arities. As in the proof of Lemma 4.8, let $\mathcal{B}^\dagger, \alpha \models \Delta_\delta$, and let Φ be the operator defined by γ. By the construction given in the proof of Lemma 4.8, $\mathcal{B}^\dagger, \alpha[x_1 \mapsto \tau_1, \ldots, x_n \mapsto \tau_n] \models \delta$ iff $(\tau_1, \ldots, \tau_n)$ is a fixed point of Φ.

We show, that Φ has only one fixed point. Let $(\tau_1, \ldots, \tau_n), (\sigma_1, \ldots, \sigma_n)$ be two fixed points of Φ. Define $\tau_i^j := \{v \in \tau_i \mid length(v) = j\}$ for any $j \geq 0$, and analogously for σ_i^j. One shows easily by induction on j that $\tau_i^j = \sigma_i^j$ for all i, j. Taking the limits of the two chains, the claim follows immediately. □

Definition 4.11 *The axiom system FX consists of the axioms (S2), (↓), (F), (A), (E) and (U).*

Corollary 4.12 *For every admissible parameter structure $\mathcal{B}$, we have that $\mathcal{B}^\dagger \models FX$.*

5 Some Properties of *FX*

5.1 Determinants

As an immediate consequence of (U) and the definition of $\exists^{\leq 1}$, we get

Proposition 5.1 *For every formula ψ and determinant δ, we have*

$$FX \models \tilde{\forall}\,(\Delta_\delta \wedge \exists \mathit{co}(\delta)\,(\delta \wedge \psi) \rightarrow \forall \mathit{co}(\delta)\,(\delta \rightarrow \psi))$$

This prominent role of determinants is the heart of the entailment check for the feature theory *CFT* (Smolka and Treinen 1994).

5.2 Primitive Formulae

Definition 5.2 *The set of* primitive formulae *is defined by the grammar*

$$p ::= \sigma \;\mid\; xt{\downarrow} \;\mid\; p \wedge p \;\mid\; p \vee p \;\mid\; \neg p \;\mid\; \forall \chi\, p \;\mid\; \exists \chi\, p$$

where σ denotes an arbitrary Σ-formula, and where χ denotes a variable not of sort Tree.

In other words, a primitive formula is a $\Sigma^\dagger$-formula that does not contain a Tree-quantifier, and does not contain an atom of the form $x \doteq y$ or $x[t]y$. A primitive formula without free Tree-variables is in fact a Σ-formula. Intuitively, in a primitive formula, the sort Tree is only used to express statements that could be as well expressed using sets. The following definition makes this intuition formal:

Definition 5.3 *We define inductively $\phi[F//x]$, the replacement of a* Tree-*variable x by a* Set-*variable F in a primitive formula ϕ.*

$$\begin{array}{rcll}
xt{\downarrow}[F//x] & = & t \mathrel{\dot\in} F & \\
a[F//x] & = & a & \textit{if } a \textit{ is an atomic formula} \\
 & & & \textit{different from } xt{\downarrow} \textit{ for all } t \\
(\neg\phi)[F//x] & = & \neg(\phi[F//x]) & \\
(\phi_1 \wedge \phi_2)[F//x] & = & \phi_1[F//x] \wedge \phi_2[F//x] & \\
(\exists \chi\, \phi)[F//x] & = & \exists \chi\,(\phi[F//x]) & \textit{if } F \neq \chi \\
(\exists F\, \phi)[F//x] & = & \exists G\,((\phi[G/F])[F//x]) & \textit{if } G \notin \mathit{fr}(\phi)
\end{array}$$

and accordingly for the derived logical junctors.

Intuitively, $\phi[F//x]$ abstracts the feature tree x in ϕ to a set F. This operation is an abstraction since it "drops" all the subtrees of a feature tree and just keeps the information about the features defined at the root. Again, this notation generalizes to simultaneous replacement $\gamma[\bar{F}//\bar{x}]$. For instance, $\phi := xa(f){\downarrow} \wedge \forall g\;(xa(g){\downarrow} \rightarrow xb(g){\downarrow})$ is a primitive formula, and

$$\phi[F//x] = a(f) \mathrel{\dot\in} F \wedge \forall g(a(g) \mathrel{\dot\in} F \rightarrow b(g) \mathrel{\dot\in} F)$$

The following lemma expresses that the definition of $\phi[F//x]$ meets the intuition of replacing a Tree-variable x by a Set-variable F.

Proposition 5.4 *Let ϕ be a primitive formula. Then*

$$\models \tilde{\forall}\left(arity(\bar{x}, \bar{F}) \rightarrow (\phi \leftrightarrow \phi[\bar{F}//\bar{x}])\right)$$

It would be possible to extend the definition of a primitive formula and of $\phi[F//x]$ to allow also for Tree-quantifiers. The definition given here is sufficient for the quantifier elimination as described below.

6 The Main Theorem

We first define the class of *restricted formulae*, which is the class of input formulae for our quantifier elimination procedure.

Definition 6.1 *[Restricted formula] A $\Sigma^\dagger$-formula is called a* restricted formula*, if in every subformula $x[t]y$ the term t is ground.*

In the following, we will also speak of restricted sentences, the restricted theory of a $\Sigma^\dagger$-structure, and so on.

Theorem 6.2 (Main Theorem) *There is an algorithm which computes for every restricted $\Sigma^\dagger$-sentence σ a Σ-sentence γ with $FX \models \sigma \leftrightarrow \gamma$.*

Before we can discuss the top-level structure of the proof, we need some additional concepts which describe the intermediate results we get during the quantifier elimination.

Definition 6.3 *[Molecule] The set of* molecules *is defined by the following grammar:*

$$m ::= x \doteq y \mid x \not\doteq y \mid x[t]y \mid \neg x[t]y \mid p$$

where p is a primitive formula, and where t is a ground term.

Hence, any molecule without free Tree-variables is in fact a primitive formula without free Tree-variables, and hence a Σ-formula.

Definition 6.4 *[Basic Formula] A* basic formula *is a $\Sigma^\dagger$-formula of the form*

$$\exists\bar{x}\,(m_1 \wedge \ldots \wedge m_n)$$

where $m_1, \ldots, m_n$ are molecules. A variable is local *to a basic formula $\exists\bar{x}\,\phi$ if it occurs in $\bar{x}$, and* global *otherwise.*

Let $\exists\bar{x}\,\phi$ be a basic formula, and let γ be the greatest graph contained in ϕ. That is, γ is the set of all molecules of the form $x[t]y$ contained in ϕ. Then we define $F^x_\phi = F^x_\gamma$.

Theorem 6.2 follows from the following lemma:

Lemma 6.5 (Main Lemma) *There is an algorithm which computes for every basic formula ϕ a universally quantified Boolean combination ψ of molecules, such that*

1. *$FX \models \tilde{\forall}\,(\phi \leftrightarrow \psi)$*
2. *$\boldsymbol{fr}(\psi) \subseteq \boldsymbol{fr}(\phi)$*
3. *if $\boldsymbol{fr}(\phi) = \emptyset$, then ψ is a boolean combination of molecules.*

We borrow the technique of proving Theorem 6.2 from Lemma 6.5 from Malc'ev 1971 and Comon and Lescanne 1989.

Proof of Theorem 6.2: It is sufficient to consider only sentences σ in a weak prenex normal form, where the matrix is just required to be boolean combination of molecules (instead of a boolean combination of atoms). We proceed by induction on the number n of quantifier blocks in the quantifier prefix.

If $n = 0$, then since σ is a sentence, it does not contain any Tree-variables and hence is a Σ-sentence.

Let $n \geq 1$ and $\sigma = Q\exists\bar{x}\ \phi$, where Q is a (possibly empty) string of quantifiers, not ending with $\exists$, and ϕ is a Boolean combination of molecules. We transform ϕ into disjunctive normal form and obtain an equivalent formula

$$Q\exists\bar{x}\ (\phi_1 \vee \ldots \vee \phi_n)$$

where every ϕ is a conjunction of molecules. This is equivalent to

$$Q(\exists\bar{x}\ \phi_1 \vee \ldots \vee \exists\bar{x}\ \phi_n)$$

where every $\exists\bar{x}\ \phi_i$ is a basic formula. Using (1) of Lemma 6.5, we can transform this equivalently into

$$Q(\forall\bar{y}_1\ \psi_1 \vee \ldots \vee \forall\bar{y}_n\ \psi_n)$$

where every ψ_i is a Boolean combination of molecules, and where all $\bar{y}_i$ are empty if Q is the empty string (because of (3) in Lemma 6.5). After possibly renaming bound variables, this can be transformed into the sentence $Q\forall\bar{z}\ \psi$, where ψ is Boolean combination of molecules. By condition (2) of Lemma 6.5, $Q\forall\bar{z}\ \psi$ is again a sentence. Since the number of quantifier alternations in $Q\forall\bar{x}\,\psi$ is $n-1$, we can now apply the induction hypothesis.

If the innermost block of quantifiers consists of universal quantifiers, we consider the negation $\neg\sigma$ of the sentence (which now has an existential innermost block of quantifiers) and transform it into a restricted sentence γ. Consequently, $FX \models \sigma \leftrightarrow \neg\gamma$. □

Corollary 6.6 *If B is an admissible Σ-structure, then the restricted theory of $B^\dagger$ is decidable relative to the theory of B.*

Note that all four admissible parameter structures introduced at the end of Section 3 have a decidable first-order theory.

1. We can interpret the theory of $\mathcal{B}_C$ in *S1S*, the monadic second order theory of natural numbers with successor. The decidability of the theory of $\mathcal{B}_C$ follows from Büchi's result (Büchi 1960) on the decidability of *S1S*.

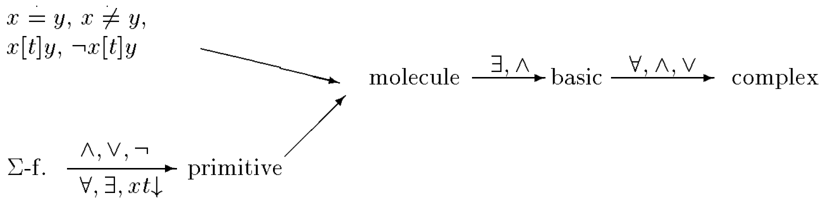

FIGURE 2 Classes of formulae

2. Analogously, the decidability of the theory of $\mathcal{B}_F$ follows from the decidability of *WS1S*, the weak monadic second order theory of the natural numbers with successor. The decidability of *WS1S* is an easy corollary of Büchi 1960, since the finite sets are definable in *S1S*.
3. Decidability of the theory of $\mathcal{B}_N$ follows again from Büchi 1960, since the initial fragments of natural numbers are definable in *S1S*.
4. Definability of the theory of $\mathcal{B}_S$ follows from Rabins celebrated result (Rabin 1969) on the decidability of *S2S*, the monadic second order theory of two successor functions. Note that the prefix relation and the lexical ordering can be defined in *S2S* (Thomas 1990).

Corollary 6.7 *The restricted theory of* $B^{\dagger}$*, where* $\mathcal{B}$ *is one of* $\mathcal{B}_C$*,* $\mathcal{B}_F$*,* $\mathcal{B}_N$*,* $\mathcal{B}_S$*, is decidable.*

7 The Reduction

We now prove Lemma 6.5. Our goal is to eliminate, by equivalence transformations with respect to FX, all the quantifiers of sort Tree, taking care not to introduce new variables. This will be achieved by transformation rules which transform basic formulae into combinations of basic formulae. To make this formal, we introduce the class of *complex formulae* (see Figure 2 for an overview of the different syntactic classes of formulae):

Definition 7.1 *[Complex formula] The set of* complex formulae *is defined by the following grammar:*

$$F ::= \forall x\ F \ \mid\ F \wedge F \ \mid\ F \vee F \ \mid\ \langle \textit{basic formula} \rangle$$

Note that this fragment, by closure of the set of molecules under negation, also contains constructions like

$$molecule_1 \wedge \ldots \wedge molecule_n \rightarrow basic\ formula$$

The transformation rules always have a basic formula in the hypothesis. Such a rule can be applied to any maximal basic formula occurring in a complex formula. The maximality condition means here, that we have to use the complete existential quantifier prefix. If a complex formula does not contain any existential quantifier then it can be easily transformed into a universally quantified boolean combination of molecules by moving universal quantifiers outside.

Definition 7.2 *[Quasi-solved from] A basic formula $\exists\bar{x}\gamma$ is a* quasi-solved form, *if*

1. *γ does not contain a molecule $x \doteq y$ or $\neg x[t]y$,*
2. *if $x \dot{\neq} y \in \gamma$, then $x \neq y$, and $x \in \bar{x}$ or $y \in \bar{x}$.*
3. *if $x[t]y \in \gamma$, then $x \in \bar{x}$,*
4. *if $x[t]y \wedge x[t]z \subseteq \gamma$, then $y = z$.*
5. *if the ground Feat-terms t, s occur in γ and $t \neq s$, then $t \dot{\neq} s \in \gamma$.*
6. *if $x \in \bar{x}$, then $\mathrm{arity}(x, F^x_\phi) \in \gamma$ or $\neg\mathrm{arity}(x, F^x_\phi) \in \gamma$.*

7.1 Transformation into Quasi-solved Form

The goal of the rules in Figure 3 is to have only basic formulae which are quasi-solved forms.

Proposition 7.3 *The rules described by (SC), (E1), (E2), (IE1), (FI) are equivalence transformations in every structure.*

Lemma 7.4 *(UD) describes an equivalence transformation in every model of the axioms schemes ($\uparrow$), (F).*

Proof: Axiom scheme (F) is equivalent to

$$\forall x, y\left(x[t]y \leftrightarrow \exists z\; x[t]z \wedge \forall z\; (x[t]z \rightarrow z \doteq y)\right)$$

which can be transformed equivalently, using axiom ($\downarrow$), into

$$\forall x, y\left(\neg x[t]y \leftrightarrow xt\!\uparrow \vee\, \exists z\; (x[t]z \wedge z \dot{\neq} y)\right)$$

As a consequence, we have for every formula ϕ with $z \notin \mathtt{fr}(\phi)$:

$$(\uparrow), (F) \models \tilde{\forall}\left(\neg x[t]y \wedge \phi \leftrightarrow (xt\!\uparrow \wedge\, \phi) \vee \exists z\; (x[t]z \wedge z \dot{\neq} y \wedge \phi)\right)$$

and hence

$$(\uparrow), (F) \models \tilde{\forall}\left(\exists\bar{x}\; (\neg x[t]y \wedge \phi) \leftrightarrow \exists\bar{x}\; (xt\!\uparrow \wedge\, \phi) \vee \exists\bar{x}, z\; (x[t]z \wedge z \dot{\neq} y \wedge \phi)\right) \quad \square$$

Lemma 7.5 *(FD) describes an equivalence transformation in every model of the axiom scheme (F).*

Lemma 7.6 *(FQ) describes an equivalence transformation in every model of the axiom scheme (F).*

Proof: We have for any formula ψ with $z \notin \mathtt{fr}(\psi)$

$$(F) \models \tilde{\forall}\left(\exists x\; (y[t]x \wedge \psi) \leftrightarrow yt\!\downarrow \wedge\, \forall z\; (y[t]z \rightarrow \psi[z/x])\right)$$

Now we choose ψ to be the formula $\exists\bar{x}\,\phi$. Since $y \notin \bar{x}$ the antecedent of the rule is equivalent to $\exists x\; (y[t]x \wedge \exists\bar{x}\; \phi)$, and the claim follows immediately. $\square$

(Sc)	$\dfrac{\exists \bar{x}\ (m \wedge \phi)}{m \wedge \exists \bar{x}\ \phi}$	$\mathtt{fr}(m) \cap \bar{x} = \emptyset$, m is not a primitive formula
(E1)	$\dfrac{\exists \bar{x}, x\ (x \doteq y \wedge \phi)}{\exists \bar{x}\ \phi[y/x]}$	$y \neq x$
(E2)	$\dfrac{\exists \bar{x}\ (x \doteq x \wedge \phi)}{\exists \bar{x}\ \phi}$	
(IE1)	$\dfrac{\exists \bar{x}\ (x \mathrel{\dot{\neq}} x \wedge \phi)}{\perp}$	
(UD)	$\dfrac{\exists \bar{x}\ (\neg x[t]y \wedge \phi)}{\begin{array}{l}\exists \bar{x}\ (xt\!\uparrow \wedge \phi) \\ \vee\ \exists \bar{x}, z\ (x[t]z \wedge z \mathrel{\dot{\neq}} y \wedge \phi)\end{array}}$	z new
(FD)	$\dfrac{\exists \bar{x}\ (x[t]y \wedge x[t]z \wedge \phi)}{\exists \bar{x}\ (x[t]y \wedge y \doteq z \wedge \phi)}$	
(FI)	$\dfrac{\exists \bar{x}\ \phi}{\begin{array}{l}s \doteq t \wedge \exists \bar{x}\ \phi[t/s] \\ \vee\ \exists \bar{x}\ (s \mathrel{\dot{\neq}} t \wedge \phi)\end{array}}$	the ground terms s, t occur in ϕ, $s \neq t$
(FQ)	$\dfrac{\exists \bar{x}, x\ (y[t]x \wedge \phi)}{yt\!\downarrow \wedge \forall z\ (y[t]z \rightarrow \exists \bar{x}\ \phi[z/x])}$	$y \notin \bar{x}$, z new

FIGURE 3 The rule set (QSF) for quasi-solved forms.

For this rule it is essential that t is ground.

Lemma 7.7 *The rule system (QSF) is terminating.*

Proof: We define a measure on basic formulae and show, that for every rule application the measure of every single basic formula generated is smaller than the measure of the basic formula being replaced. Termination then follows by a standard multiset argument. We assign a basic formula γ the tuple $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, where

1. α_1 is the number of $\neg x[t]y$ molecules in γ,
2. α_2 is the number of $x[t]y$ molecules in γ,
3. α_3 is the number of pairs (t, s) of Feat-ground terms, where both t and s occur in γ, but $t \dot{\neq} s$ does not occur in γ,
4. α_4 is the total length of γ.

It is now easily checked that the lexicographic ordering on these measures is strictly decreased by every application of a rule. The side condition of rule (Sc) guarantees that no formula of the form $t \dot{\neq} s$, $arity(x, F_\phi^x)$ or $\neg arity(x, F_\phi^x)$ is moved out of a basic formula. □

Corollary 7.8 *There is an algorithm, which transforms any basic formula into an FX-equivalent complex formula, in which all basic formulae are quasi-solved forms.*

Proof: We compute a normal-from wrt. the ruleset (QSF), and from this compute a normal form wrt. the following rule:

$$\text{(ST)} \quad \frac{\exists \bar{x}, x\, \phi}{\begin{array}{c}\exists \bar{x}, x\, (\phi \wedge arity(x, F_\phi^x)) \\ \vee\ \exists \bar{x}, x\, (\phi \wedge \neg arity(x, F_\phi^x))\end{array}} \quad arity(x, F_\phi^x), \neg arity(x, F_\phi^x) \notin \phi$$

□

7.2 Eliminating Quasi-Solved Forms with Sloppy Inequations

In this section, we show how to eliminate quasi-solved forms with only benign inequations, in a sense to be explained soon. In the next subsection, we will explain how to get rid of nasty inequations.

Definition 7.9 *[Sloppy and Tight variables] Let $\exists \bar{x}\, \gamma$ be a basic formula. We call a local variable $x \in \bar{x}$* tight *(in $\exists \bar{x}\, \gamma$) if $arity(x, F_\gamma^x) \in \gamma$, and otherwise* sloppy*.*

By the definition of a quasi-solved form, $\neg arity(x, F_\phi^x) \in \gamma$ for every sloppy variable x.

Definition 7.10 *[Closure] For a graph γ, we define for every feature path π of* Feat*-terms the relation $\rightsquigarrow_\gamma^\pi$ as the smallest relation on $\mathit{fr}(\gamma)$ with*

$$x \rightsquigarrow_\gamma^\epsilon x \quad \text{if } x \in \mathit{fr}(\gamma)$$

$$\text{if } x \rightsquigarrow_\gamma^\pi y \text{ and } y[t]z \in \gamma, \text{ then } x \rightsquigarrow_\gamma^{\pi t} z$$

We write $x \rightsquigarrow_\gamma y$ if $x \rightsquigarrow_\gamma^\pi y$ for some π.

For a graph γ and variables x, y, we define the closure *of (x, y)*

$$\langle x, y\rangle_\gamma := \{(u, v) \in \mathit{fr}(\gamma)^2 \mid x \rightsquigarrow_\gamma^\pi u \text{ and } y \rightsquigarrow_\gamma^\pi v \text{ for some } \pi\}$$

In Backofen 1995, the variable y with $x \rightsquigarrow_\gamma^\pi y$ has been called the *value* $|x\pi|_\gamma$ of the rooted path $x\pi$ in γ. Obviously, $\langle x, y\rangle_\gamma$ can be computed in finitely many steps.

Proposition 7.11 *For every graph γ, variables x, y and $(u, v) \in \langle x, y\rangle_\gamma$ we have*

$$(F) \models (\gamma \wedge u \not\doteq v \rightarrow x \not\doteq y)$$

Definition 7.12 *[Sloppy and Tight inequations] Let $\exists\bar{x}\,\gamma$ be a basic formula. We call an inequation $x \not\doteq y$* sloppy (in $\exists\bar{x}\,\gamma$)*, if there is a $(u, v) \in \langle x, y\rangle_\gamma$ with $x \neq y$, where at least one of u and v is sloppy. Otherwise, the inequation is called* tight.

The benign inequations handled in this section are the sloppy ones. The idea is that for sloppy variables, we have enough freedom to make them all different.

In the following, we assume a partition of a quasi-solved form as $\exists\bar{x}\,(\gamma \wedge \iota \wedge \rho)$, where γ denotes a graph, ι denotes a conjunction of inequations between Tree-variables, and ρ denotes a primitive formula. Note that in this case, by the definition of quasi-solved forms, $\mathtt{co}(\gamma) \subseteq \bar{x}$, $\Delta_\gamma \subseteq \rho$, and ι contains only non-trivial inequations which use at least one local variable. For a graph γ, we denote by $\tilde{\gamma}$ the formula obtained by replacing every atom $x[t]y$ by $xt{\downarrow}$.

Lemma 7.13 *Let $\exists\bar{x}\,(\gamma \wedge \rho)$ be a quasi solved form without inequations. Then*

$$FX \models \tilde{\forall}\left(\exists\bar{x}\,(\gamma \wedge \rho) \rightarrow \exists\bar{F}\,((\tilde{\gamma} \wedge \rho)[\bar{F}//\bar{x}])\right)$$

where $\bar{F}$ is disjoint with $\mathit{fr}(\rho)$.

Proof: Let $\mathcal{A} \models FX$ and α be a valuation with $\mathcal{A}, \alpha \models \exists\bar{x}\,(\gamma \wedge \rho)$. Since $\models \gamma \rightarrow \tilde{\gamma}$, we get $\mathcal{A}, \alpha \models \exists\bar{x}\,(\tilde{\gamma} \wedge \rho)$. Together with axiom (A), this means since $\bar{F}$ is disjoint with $\mathtt{fr}(\rho)$, that $\mathcal{A}, \alpha \models \exists\bar{x}, \bar{F}\,(\mathit{arity}(\bar{x}, \bar{F}) \wedge \tilde{\gamma} \wedge \rho)$. With Proposition 5.4, we get

$$\mathcal{A}, \alpha \models \exists\bar{F}\,((\tilde{\gamma} \wedge \rho)[\bar{F}//\bar{x}])$$

since $\bar{x}$ is disjoint with $\mathtt{fr}((\tilde{\gamma} \wedge \rho)[\bar{F}//\bar{x}])$. $\square$

Lemma 7.14 *Let $\exists\bar{x}\,(\gamma \wedge \iota \wedge \rho)$ be a quasi solved form, where $\bar{F}$ is disjoint with $\mathtt{fr}(\rho)$ and ι consists of sloppy inequations only. Then*

$$FX \models \tilde{\forall}\left(\exists\bar{F}\;((\tilde{\gamma} \wedge \rho)[\bar{F}/\!/\bar{x}]) \rightarrow \exists\bar{x}\;(\gamma \wedge \iota \wedge \rho)\right)$$

Proof: Let $\mathcal{A} \models FX$ and α be a valuation with $\mathcal{A}, \alpha \models \exists\bar{F}\;(\tilde{\gamma} \wedge \rho)[\bar{F}/\!/\bar{x}]$. Let β be an $\bar{F}$-update of α, such that $\mathcal{A}, \beta \models (\tilde{\gamma} \wedge \rho)[\bar{F}/\!/\bar{x}]$. Let Sl be the set of sloppy variables of $\gamma \wedge \rho$. Let f, F be new variables, and for every $x \in Sl$, let $n_x \geq 0$, and $f_x, x^0, \ldots, x^{n_x}$ be variables not occurring in $\gamma \wedge \rho$. Let $Slf = \{f_x \mid x \in Sl\}$, and $Slx = \{x^i \mid x \in Sl, 0 \leq i \leq n_x\}$. We define an extension ξ of γ by

$$\xi := \gamma \wedge \bigwedge_{x \in Sl} (x[f_x]x^0 \wedge x^0[f]x^1 \wedge \ldots x^{n_x - 1}[f]x^{n_x} \wedge \mathit{arity}(x^{n_x}, F))$$

Hence, $\models \xi \rightarrow \gamma$. By axiom (S2), there are $a \in \mathsf{Feat}^{\mathcal{A}}$ and $A, B \in \mathsf{Set}^{\mathcal{A}}$ with $a \in^{\mathcal{A}} A$ and $a \notin^{\mathcal{A}} B$. We denote by $\bar{\bar{x}}$ the extension of $\bar{x}$ by Slx, and by $\bar{\bar{F}}$ an according extension of $\bar{F}$. Hence, by definition of sloppyness, there is a $Slf \cup Slx \cup \{f, F\}$-update β' of β such that

$$\mathcal{A}, \beta' \models \Delta_\xi \wedge (\tilde{\xi} \wedge \rho)[\bar{\bar{F}}/\!/\bar{\bar{x}}]$$

Especially, $\beta'(f) = a$, $\beta'(F^i) = A$ if F corresponds to some x^i with $i \leq n_x$, and $\beta'(F^i) = B$ if F corresponds to some x^{n_x}. Note, that Δ_ξ extends $\Delta_\gamma \subseteq \rho$ just by stating that f_x is assigned a value different from all (ground) terms in F^x_γ. By construction, $\mathcal{A}, \beta' \models \bigwedge_{i=1}^{n} \bigwedge_{a \in F_\xi^{x_i}} a \in F_i$. Hence, by axiom (E), there is an $\bar{\bar{x}}$-update β'' of β', such that

$$\mathcal{A}, \beta'' \models \xi \wedge \mathit{arity}(\bar{\bar{x}}, \bar{\bar{F}})$$

Let $\alpha'(x) = \beta''(x)$ if $x \in \bar{x}$, and $\alpha'(x) = \alpha(x)$ otherwise. Hence, $\mathcal{A}, \alpha' \models \gamma$. By Proposition 5.4 and since $\bar{F}$ is disjoint with $\mathtt{fr}(\rho)$, $\mathcal{A}, \alpha' \models \rho$.

Since there are infinitely many choices of n_x for every $x \in Sl$, we can easily find values n_x such that $\beta''(x) \neq \beta''(y)$ for every variable $y \in \mathtt{fr}(\gamma \wedge \iota \wedge \rho)$ with $y \neq x$. Hence, by Proposition 7.11, $\mathcal{A}, \alpha' \models \iota$. □

We are now ready to give the elimination rule for quasi-solved forms with benign inequations:

$$\text{(IE2)} \quad \frac{\exists\bar{x}\;(\gamma \wedge \iota \wedge \rho)}{\exists\bar{F}\;((\tilde{\gamma} \wedge \rho)[\bar{F}/\!/\bar{x}])} \quad \begin{array}{l}\text{if } \iota \text{ contains only sloppy inequations} \\[2ex] \bar{F} \cap \mathtt{fr}(\rho) = \emptyset\end{array}$$

As an example of rule (IE2), consider

$$\frac{\begin{array}{l}\exists x, y, u\ (\ x[s]y \wedge u[s]v \wedge y[t]y \wedge x \neq u \wedge arity(x, \{s\}) \wedge \\ \qquad arity(u, \{s\}) \wedge \neg arity(y, \{t\}))\end{array}}{\begin{array}{l}\exists F, G, H\ (\ s \mathrel{\dot\in} F \wedge s \mathrel{\dot\in} G \wedge t \mathrel{\dot\in} H \wedge \forall \nu\, (\nu \mathrel{\dot\in} F \leftrightarrow \nu \doteq s) \wedge \\ \qquad \forall \nu\, (\nu \mathrel{\dot\in} G \leftrightarrow \nu \doteq s) \wedge \neg \forall \nu\, (\nu \mathrel{\dot\in} F \leftrightarrow \nu \doteq s))\end{array}}$$

Here, $x \neq u$ is a sloppy inequation since y is a sloppy variable.

From Lemma 7.14 and Lemma 7.13, we get immediately

Lemma 7.15 *The rule (IE2) describes an equivalence transformation in every model of FX.*

Corollary 7.16 *There is an algorithm, which transforms any complex formula, in which all basic formulae are quasi-solved forms containing only sloppy inequations, into an FX-equivalent universally quantified boolean combination of molecules.*

7.3 Eliminating Tight Inequations

In the closure of tight inequations, there are only inequations of type *tight* $\neq$ *tight* or *tight* $\neq$ *global*. We first show how to transform the quasi-solved form such that the only tight inequations are of type *tight* $\neq$ *global*. Then, we show how to get rid of the *tight* $\neq$ *global* inequations.

$$\text{(IE3)}\quad \frac{\exists \bar{x}\, (\gamma \wedge \iota \wedge x \neq y \wedge \rho)}{\exists \bar{x}\, (\gamma \wedge \iota \wedge \rho)} \quad \begin{array}{l}\text{there are tight variables } u, v \text{ with} \\ (u, v) \in \langle x, y\rangle_\gamma \text{ and } F^u_\gamma \neq F^v_\gamma\end{array}$$

From Proposition 7.11, we get

Proposition 7.17 *The rule (IE3) describes an equivalence transformation on quasi-solved forms in every model of FX.*

Proof: This is a consequence of condition (5) in the definition of a quasi-solved form. □

We say that the set η of equations is *closed* under a graph γ, if whenever $x \doteq y \in \gamma$ and $(u, v) \in \langle x, y\rangle_\gamma$, then $u \doteq v \in \gamma$.

Proposition 7.18 *Let δ be a determinant and η a set of equations which is closed under δ. If $\mathit{fr}(\eta) \subseteq \mathtt{co}(\delta)$ and $F^x_\delta = F^y_\delta$ for every equation $x \doteq y \in \eta$, then*

$$FX \models \tilde\forall\, (\Delta_\delta \rightarrow (\delta \rightarrow \eta))$$

Proof: Let $\mathcal{A}, \alpha \models \Delta_\delta$. By Proposition 5.1, we have to show that

$$(8) \qquad \mathcal{A}, \alpha \models \exists \mathtt{co}(\delta)(\delta \wedge \eta))$$

Let θ be an idempotent substitution equivalent to η. Then

$$(8) \quad \Leftrightarrow \quad \mathcal{A}, \alpha \models \exists \mathtt{co}(\delta)(\delta \wedge \theta)$$

$$\Leftrightarrow \quad \mathcal{A}, \alpha \models \exists \mathtt{co}(\delta)(\theta\delta \wedge \theta)$$

(9) $$\Leftrightarrow \quad \mathcal{A}, \alpha \models \exists \mathtt{co}(\delta)(\theta\delta) \qquad \text{since } \mathtt{fr}(\theta) \subseteq \mathtt{co}(\delta)$$

By construction, $\theta\delta$ is again a determinant, with $\mathtt{co}(\theta) \subseteq \mathtt{co}(\delta)$, and $\Delta_{\theta\delta} = \Delta_\delta$. Hence, (9) follows from axiom (E). □

A similar lemma, in the context of CFT, was presented in Smolka and Treinen 1994.

Proposition 7.19 *Let δ be a determinant and η, η' be sets of equations such that $\eta \wedge \eta'$ is closed under δ. If $\mathtt{fr}(\eta) \subseteq \mathtt{co}(\delta)$ and $F^x_\delta = F^y_\delta$ for every equation $x \doteq y \in \eta$, then*

$$FX \models \Delta_\delta \rightarrow \tilde{\forall}\,(\delta \rightarrow (\eta' \leftrightarrow \eta \wedge \eta'))$$

Proof: We have to show that

(10) $$FX \models \tilde{\forall}\,(\Delta_\delta \wedge \delta \wedge \eta' \rightarrow \eta)$$

Let θ' be an idempotent substitution equivalent to η'. Then (10) is equivalent to

(11) $$FX \models \tilde{\forall}\,(\Delta_\delta \wedge \theta'\delta \rightarrow \theta'\eta)$$

since $\theta'\Delta_\delta = \Delta_\delta$. Observe, that $\Delta_\delta = \Delta_{\theta'\delta}$, $\mathtt{fr}(\theta'\eta) \subseteq \mathtt{co}(\theta'\delta)$, $\theta'\eta$ is closed under $\theta'\delta$, and that $F^x_{\theta'\delta} = F^y_{\theta'\delta}$ for every equation $x \doteq y \in \gamma$. Hence, (11) follows from Proposition 7.18. □

We can now give the rule which reduces the *tight* $\neq$ *tight* inequations to *tight* $\neq$ *global* inequations:

(IE4) $$\frac{\exists \bar{x}\,(\gamma \wedge \iota \wedge x \dot{\neq} y \wedge \rho)}{\bigvee_{(u,v)\in I} \exists \bar{x}\,(\gamma \wedge \iota \wedge u \dot{\neq} v \wedge \rho)}$$ $x \dot{\neq} y$ tight, (IE3) does not apply, $I = \{(u,v) \in \langle x, y\rangle_\gamma \mid \{u,v\} \not\subseteq \bar{x}\}$

As an example of rule (IE4), consider

$$\frac{\begin{array}{ll} \exists x, y, v\,(& x[s]v \wedge x[t]v' \wedge y[s]w \wedge y[t]w' \wedge s \dot{\neq} t \wedge \\ & arity(x, \{s,t\}) \wedge arity(y, \{s,t\}) \wedge arity(v, \{\}) \wedge x \dot{\neq} y) \end{array}}{\begin{array}{lll} & \exists x, y, v\,(& x[s]v \wedge x[t]v' \wedge y[s]w \wedge y[t]w' \wedge s \dot{\neq} t \wedge \\ & & arity(x, \{s,t\}) \wedge arity(y, \{s,t\}) \wedge arity(v, \{\}) \wedge v \dot{\neq} w) \\ \vee & \exists x, y, v\,(& x[s]v \wedge x[t]v' \wedge y[s]w \wedge y[t]w' \wedge s \dot{\neq} t \wedge \\ & & arity(x, \{s,t\}) \wedge arity(y, \{s,t\}) \wedge arity(v, \{\}) \wedge v' \dot{\neq} w') \end{array}}$$

Lemma 7.20 *The rule (IE4) describes an equivalence transformation in every model of FX.*

Proof: This follows immediately from Proposition 7.19. □

Finally, we give the rule to eliminate *tight* $\neq$ *global* inequations.

Definition 7.21 *[Generated subformula] For a conjunction ϕ of molecules and variable x, the* subformula ϕ_x of ϕ generated by x *is defined as*

$$\phi_x := \{u[t]v, arity(u, M) \in \phi \mid x \rightsquigarrow_\phi u\}$$

Note that, if $x \dot{\neq} y$ is tight in the quasi-solved form $\exists \bar{x}\, \phi$, then ϕ_x is a determinant.

$$\text{(IE5)} \quad \frac{\exists \bar{x}, x\ (\phi \wedge x \dot{\neq} y)}{\exists \bar{x}, x\ \phi \wedge \forall \mathsf{co}(\phi_x)\ (\phi_x \rightarrow x \dot{\neq} y)} \quad y \notin \bar{x}, y \neq x, x \text{ tight}$$

As an example of rule (IE5), consider

$$\frac{\exists x, x'\ (x[s]x \wedge x[t]y \wedge x'[t]x' \wedge s \dot{\neq} t \wedge arity(x, \{f\}) \wedge x \dot{\neq} y)}{\begin{array}{l}\exists x, x'\ (x[s]x \wedge x[t]y \wedge x'[t]x' \wedge s \dot{\neq} t \wedge arity(x, \{f\})) \wedge \\ \forall x\ (x[s]x \wedge x[t]y \wedge arity(x, \{f\}) \rightarrow x \dot{\neq} y)\end{array}}$$

Lemma 7.22 *The rule (IE5) describes an equivalence transformation in every model of FX.*

Proof: First note that $\mathsf{co}(\phi_x) \subseteq \bar{x} \cup \{x\}$. Since $\models \phi \rightarrow \phi_x$, the conclusion implies the hypothesis.

The hypothesis obviously implies the first part of the conclusion. By Proposition 5.1, it also implies the second part (note that $\Delta_{\phi_x} \subseteq \phi$, since ϕ is a quasi-solved form). □

Corollary 7.23 *There is an algorithm, which transform any complex formula in which all basic formulae are quasi-solved forms, into an FX-equivalent complex formula, in which all basic formulae are quasi-solved forms containing only sloppy inequations.*

Hence, we obtain the proof of Lemma 6.5 by composing the Corollaries 7.8, 7.16 and 7.23.

Acknowledgments. David Israel pointed out the analogy to the situation in process logic. Rolf Backofen, Patrick Blackburn, Andreas Podelski and Gert Smolka provided helpful criticism and remarks.

This work has been partially supported by the Bundesminister für Bildung, Wissenschaft, Forschung und Technologie (Hydra, ITW 9105), the Human Capital and Mobiliy Programme of the European Union (SOL, CHRX-CT92-0053 and CONSOLE, CHRX-CT91-0195), the Esprit Basic Research Project ACCLAIM (EP 7195) and the Esprit Working Group CCL (EP 6028).

References

Aït-Kaci, Hassan. 1986. An Algebraic Semantics Approach to the Effective Resolution of Type Equations. *Theoretical Computer Science* 45:293–351.

Aït-Kaci, Hassan, and Roger Nasr. 1986. LOGIN: A Logic Programming Language with Built-In Inheritance. *Journal of Logic Programming* 3:185–215.

Aït-Kaci, Hassan, and Andreas Podelski. 1991. Towards a Meaning of LIFE. In *3rd International Symposium on Programming Language Implementation and Logic Programming*, ed. Jan Maluszyński and Martin Wirsing, 255–274. Lecture Notes in Computer Science, vol. 528. Springer-Verlag, August.

Aït-Kaci, Hassan, Andreas Podelski, and Gert Smolka. 1994. A Feature-based Constraint System for Logic Programming with Entailment. *Theoretical Computer Science* 122(1–2):263–283.

Andrews, Peter B. 1986. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof.* Computer Science and Applied Mathematics. Academic Press.

Backofen, Rolf. 1994a. *Expressivity and Decidability of First-Order Theories over Feature Trees.* Doctoral dissertation, Technische Fakultät der Universität des Saarlandes, Saarbrücken, Germany.

Backofen, Rolf. 1994b. Regular Path Expressions in Feature Logic. *Journal of Symbolic Computation* 17:421–455.

Backofen, Rolf. 1995. A Complete Axiomatization of a Theory with Feature and Arity Constraints. *Journal of Logic Programming* 24(1&2):37–71.

Backofen, Rolf, and Gert Smolka. 1995. A Complete and Recursive Feature Theory. *Theoretical Computer Science* 146(1–2):243–268.

Backofen, Rolf, and Ralf Treinen. 1994. How to Win a Game with Features. In *1st International Conference on Constraints in Computational Logics*, ed. Jean-Pierre Jouannaud. Lecture Notes in Computer Science, vol. 845. München, Germany, September. Springer-Verlag. Extended Version accepted for publication in Information and Control, special issue on CCL'94.

Büchi, J. R. 1960. On a Decision Method in Restricted Second Order Arithmetic. In *International Congr. on Logic, Methodology and Philosophy of Science*, ed. E. Nagel et. al., 1–11. Stanford University Press.

Comon, Hubert, and Pierre Lescanne. 1989. Equational Problems and Disunification. *Journal of Symbolic Computation* 7(3,4):371–425.

Dörre, Jochen. 1993. *Feature-Logik und Semiunfikation.* Doctoral dissertation, Philosophische Fakultät der Universität Stuttgart, July. In German.

Dörre, Jochen. 1994. Feature-Logic with Weak Subsumption Constraints. In *Constraints, Language and Computation*, ed. M. A. Rosner C. J. Rupp and R. L. Johnson. Chap. 7, 187–203. Academic Press.

Dörre, Jochen, and William C. Rounds. 1992. On Subsumption and Semiunification in Feature Algebras. *Journal of Symbolic Computation* 13(4):441–461.

Gunter, C. A., and D. S. Scott. 1990. Semantic Domains. In van Leeuwen van Leeuwen 1990, Chap. 12, 633–674.

Henz, Martin, Gert Smolka, and Jörg Würtz. 1995. Object-oriented Concurrent Constraint Programming in Oz. In *Principles and Practice of Constraint*

Programming, ed. V. Saraswat and P. Van Hentenryck. Chap. 2, 27–48. Cambridge, MA: MIT Press. To appear.

Hodges, Wilfrid. 1993. *Model Theory*. Encyclopedia of Mathematics and its Applications 42. Cambridge University Press.

Jaffar, Joxan, and Jean-Louis Lassez. 1987. Constraint Logic Programming. In *Proceedings of the 14th ACM Conference on Principles of Programming Languages*, 111–119. Munich, Germany, January. ACM.

Johnson, Mark. 1988. *Attribute-Value Logic and the Theory of Grammar*. CSLI Lecture Notes 16. Stanford University, CA: Center for the Study of Language and Information.

Malc'ev, Anatolií Ivanovič. 1971. Axiomatizable Classes of Locally Free Algebras of Various Type. In *The Metamathematics of Algebraic Systems: Collected Papers 1936–1967*, ed. III Benjamin Franklin Wells. Chap. 23, 262–281. North Holland.

Parikh, Rohit. 1978. A Decidability Result for a Second Order Process Logic. In *19th Annual Symposion on Foundations of Computer Science*, 177–183. Ann Arbor, Michigan, October. IEEE.

Rabin, Michael O. 1969. Decidability of Second-Order Theories and Automata on Infinite Trees. *Transactions of the American Mathematical Society* 141:1–35.

Rounds, William C., and Robert Kasper. 1986. A Complete Logical Calculus for Record Structures Representing Linguistic Information. In *Proceedings of the First Symposium on Logic in Computer Science*, 38–43. Cambridge, MA, June. IEEE Computer Society.

Smolka, Gert. 1992. Feature Constraint Logics for Unification Grammars. *Journal of Logic Programming* 12:51–87.

Smolka, Gert. 1995. The Oz Programming Model. In *Current Trends in Computer Science*, ed. Jan van Leeuwen. 324–343. Lecture Notes in Computer Science, vol. 1000. Berlin, Heidelberg, New York: Springer-Verlag.

Smolka, Gert, and Hassan Aït-Kaci. 1989. Inheritance Hierarchies: Semantics and Unification. *Journal of Symbolic Computation* 7:343–370.

Smolka, Gert, and Ralf Treinen. 1994. Records for Logic Programming. *Journal of Logic Programming* 18(3):229–258.

Thomas, Wolfgang. 1990. Automata on Infinite Objects. In van Leeuwen van Leeuwen 1990, Chap. 4, 133–191.

Treinen, Ralf. 1992. A New Method for Undecidability Proofs of First Order Theories. *Journal of Symbolic Computation* 14(5):437–457.

Treinen, Ralf. 1993. Feature Constraints with First-Class Features. In *Mathematical Foundations of Computer Science*, ed. Andrzej M. Borzyszkowski and Stefan Sokołowski, 734–743. Lecture Notes in Computer Science, vol. 711. Springer-Verlag, 30 August–3 September.

van Leeuwen, Jan (ed.). 1990. *Handbook of Theoretical Computer Science*. Elsevier Science Publishers and The MIT Press.