Constraints, , 1–33 ()

Ordering Constraints over Feature Trees

MARTIN MÜLLER http://www.ps.uni-sb.de/~mmueller

JOACHIM NIEHREN http://www.ps.uni-sb.de/~niehren

Programming Systems Lab, Universität des Saarlandes,
Stuhlsatzenhausweg 3, D-66041 Saarbrücken, Germany

ANDREAS PODELSKI http://www.mpi-sb.mpg.de/~podelski

Max-Planck-Institut für Informatik,
Im Stadtwald, D-66123 Saarbrücken, Germany

Received April 14, 1998; Revised September 6, 1998

Editor: Gert Smolka

Abstract. Feature trees are the formal basis for algorithms manipulating record like structures in constraint programming, computational linguistics and in concrete applications like software configuration management. Feature trees model records, and constraints over feature trees yield extensible and modular record descriptions. We introduce the constraint system $\mathrm{FT}_{\leq}$ of ordering constraints interpreted over feature trees. Under the view that feature trees represent symbolic information, the relation $\leq$ corresponds to the information ordering ("carries less information than"). We present two algorithms in cubic time, one for the satisfiability problem and one for the entailment problem of $\mathrm{FT}_{\leq}$. We show that $\mathrm{FT}_{\leq}$ has the independence property. We are thus able to handle negative conjuncts via entailment and obtain a cubic algorithm that decides the satisfiability of conjunctions of positive and negated ordering constraints over feature trees. Furthermore, we reduce the satisfiability problem of Dörre's weak subsumption constraints to the satisfiability problem of $\mathrm{FT}_{\leq}$ and improve the complexity bound for solving weak subsumption constraints from $O(n^5)$ to $O(n^3)$.

Keywords: feature constraints, tree orderings, weak subsumption, satisfiability, entailment, complexity

1. Introduction

Feature logic is a formalism for describing record structures, which in turn represent objects – such as addresses or lexical entries – by the values of their attributes. Feature logic has its origin in the three areas of knowledge representation with *concept descriptions*, *frames*, or ψ-terms [13, 34, 35, 1], natural language processing, in particular approaches based on *unification grammars* [26, 24, 45, 43, 39, 42], and constraint (logic) programming [3, 5, 28, 47]. An interesting recent application lies in software configuration management, where feature logic is used to denote software versions and to deduce their mutual consistency [50, 51].

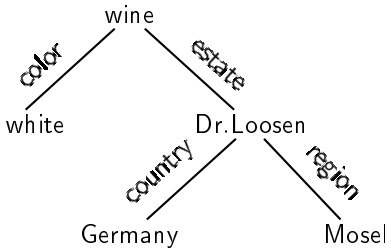
The first mathematical treatment of record descriptions was the formalisms of ψ-*terms* [1]. In other approaches, ψ-terms were called *feature structures* [40] or *feature terms* [46]. In contrast to earlier work, the notion *feature structure* was mostly used for designating a record structure itself [14, 39, 42] rather than a record description. Logical descriptions of record structures lead to the notion of *feature logic* [25, 23, 46]. When we call these descriptions *feature constraints*, feature unification becomes constraint solving.

Two main approaches to feature logics should be clearly distinguished. In computational linguistics [14, 39, 42], a record structure is traditionally described from an internal perspective, i.e. by specifying relationships between its nodes. Motivated by constraint programming, Smolka proposed an alternative approach [6, 11, 49, 48, 7, 9] based on an external view in which record structures are described by relations to others. The internal view was modeled conveniently in terms of *feature graphs* and constraints with variables for nodes of a feature graph. In contrast, the external view lead to the notion of *feature trees* – instead of more general feature graphs – and feature constraints with variables for feature trees. Nevertheless, logical theories based on these two different views often turned out to be elementarily equivalent [7, 11, 9].

In this article, we follow the external view based on the notion of feature trees. We introduce and investigate the constraint system $FT_{\leq}$ of ordering constraints over feature trees which extends the system FT of equality constraints over feature trees [6, 11]. Before presenting these constraint languages, we discuss feature trees and their ordering.

A *feature tree* is a tree with unordered, labeled edges and labeled nodes. The edge labels are called features; features are functional in that each two features labeling edges departing from the same node are distinct.
In programming, features correspond to record field selectors and node labels to record field contents.

An example of a feature tree is displayed on the right. Its root is labeled with the node label wine and the edges departing at is root are labeled by the features color and estate.

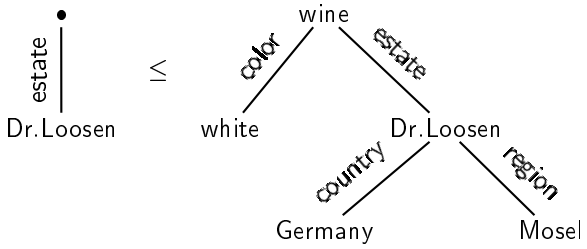


A feature tree is defined by a tree domain and a labeling function. The domain of a feature tree τ is the set all words labeling a branch from the root of τ to a node of τ. For instance, the domain of the above tree is $\{\varepsilon, \mathsf{color}, \mathsf{estate}, \mathsf{estate\ country}, \mathsf{estate\ region}\}$.

A feature tree is finite if its tree domain is finite. In general, the domain of a feature tree may also be infinite in order to model records with cyclic dependencies. Notice that every ground term such as $\mathsf{square}(\mathsf{plus}(\mathsf{a},\mathsf{b}))$ can be considered as a finite feature tree where the features are just consecutive natural numbers.

A feature tree can be seen as a carrier of information. This viewpoint gives rise to an ordering relation on feature trees in a very natural way that we call *information ordering*. In the framework of feature algebras the same ordering was called weak subsumption ordering [19].

The information ordering is illustrated by the example to the right: The smaller tree is missing some information about the object it represents,



namely that this object is a white wine and that the estate of Dr. Loosen is located at the Mosel in Germany. In order to have nodes without information, we allow for unlabeled nodes depicted with a $\bullet$. Formally, this means that we do not require a labeling function to be total.

Intuitively, a feature tree τ_1 is smaller than a feature tree τ_2 if τ_1 has fewer edges and node labels than τ_2. More precisely, this means that every word of features in the tree domain of τ_1 belongs to the tree domain of τ_2 and that the partial labeling function of τ_1 is contained in the labeling function of τ_2. In this case we write $\tau_1 \leq \tau_2$.

The feature constraints in the constraint systems FT [6, 11] are conjunctions of three kinds of atomic formulas which are built from variables x, features f and node labels a:

$x{=}y$ ("the trees x and y have the same structure"),

$a(x)$ ("the root of x is labeled a"),

$x[f]x'$ ("x' is the subtree of x accessed via the edge labeled with f.").

For instance, the larger tree depicted above is a possible value for x in a solution of the constraint $\mathsf{wine}(x) \wedge x[\mathsf{estate}]x_1 \wedge x_1[\mathsf{region}]x_2$ but the smaller one is not. Feature constraints in FT are modular and extensible in that pieces of information can be added feature by feature. Note also that no constraint in FT can uniquely determine a single feature tree. For instance, there is no way to express in FT that a feature tree has no edges at all. This can, however, be stated in CFT by using arity constraints [48, 12].

In the constraint system FT, we may constrain the values for x and y to be equal, $x = y$. In some situations (*e.g.*, in computational linguistics; see below), we may need a weaker constraint on x and y. We may want to express, for example, that y represents at least the information of x (but possibly more), formally $x \leq y$. Or, we may want to say that x and y express compatible information, formally $x{\sim}y$. Since this is equivalent to saying that there exists a common refinement of the information of x and y, formally: $\exists z(x{\leq}z \wedge y{\leq}z)$, compatibility $\sim$ can be reduced to the information ordering $\leq$.

In this article, we introduce the constraint system $\mathrm{FT}_{\leq}$ of information ordering constraints over feature trees. We obtain the system $\mathrm{FT}_{\leq}$ from FT [6] by replacing equalities $x{=}y$ by more general ordering constraints $x{\leq}y$. The abstract syntax of ordering constraints φ in $FT_{\leq}$ is defined as follows where x and x' are variables, f a *feature* and a a *label*.

$$\varphi \ ::= \ x{\leq}x' \ \mid \ x[f]x' \ \mid \ a(x) \ \mid \ \varphi \wedge \varphi'$$

The semantics of ordering constraints is given by interpretation over feature trees where the symbol $\leq$ is interpreted as information ordering. Throughout the paper, we consider two cases, either we interprete in the structure of finite feature trees or else in the structure of arbitrary feature trees.

In contrast with the situation in previous feature constraint systems [48, 6, 9], the nodes of a feature tree in the interpretation domain of $\mathrm{FT}_{\leq}$ are possibly unlabeled. This fact is insignificant when only equality constraints are considered: The first-order theory of FT does not change when the structure allows for partially labeled feature trees [11][1]. In contrast, when ordering constraints $x{\leq}y$ are involved, this choice is significant. The first-order theories of ordering constraints interpreted in the structure of partially labeled feature

trees differ from the one interpreted in the structure of completely labeled feature trees. For instance, the constraint $\exists z(z{\leq}x \wedge z{\leq}y)$ is valid over partially labeled feature trees but not over completely labeled trees. This accounts for the fact that the information ordering $\leq$ has a least element, the tree with a single unlabeled node. The choice of partially labeled trees in the semantics of $\mathrm{FT}_{\leq}$ has algorithmic consequences as well; *i.e.*, the correctness of our algorithms depends on it. There also exists a natural extension of the notion of partially labeled feature trees in terms of completely labeled feature trees with a partial ordering on node labels [29].

It is clear that $\mathrm{FT}_{\leq}$ is as expressive as FT since its ordering is antisymmetric. We formally prove that $\mathrm{FT}_{\leq}$ is strictly more expressive than FT by showing that no constraint in FT can be equivalent to $x{\leq}x'$.

We present two cubic time algorithms for $\mathrm{FT}_{\leq}$, one which solves its satisfiability problem ("Is φ satisfiable in $\mathrm{FT}_{\leq}$?") and one which solves its entailment problem ("Is $\varphi \rightarrow \varphi'$ valid in $\mathrm{FT}_{\leq}$?"). Note carefully that entailment (in contrast to satisfiablity) becomes much harder if arity constraints or existential quantification are added to the constraint language $FT_{\leq}$ [33].

The satisfiability test for $FT_{\leq}$ can be applied for type inference with record types or object types [37], but also for the syntactical treatment of coordination phenomena in natural language processing [19, 39]. The entailment test might be useful for constraint simplification [41] during record type inference, but it is also prerequisite for a possible usage of $FT_{\leq}$ in modern constraint programming languages with advanced control mechanisms such as delaying, coroutining, synchronization, committed choice and local computation spaces [2, 5, 47, 38].

We furthermore show that $\mathrm{FT}_{\leq}$ has the independence property if the set of features provided by the signature is infinite. Thanks to the independence property, the entailment test is sufficient for testing conjunctions $\varphi \wedge \neg\varphi_1 \wedge \ldots \wedge \neg\varphi_n$ for satisfiability (namely, by testing that none of the judgments $\varphi \models \varphi_i$ holds for all $1 \leq i \leq n$). We are thus able to handle negative conjuncts via entailment. We can summarize our algorithmic results by saying that the satisfiability problem of conjunctions of positive and negative ordering constraints $\varphi \wedge \neg\varphi_1 \wedge \ldots \wedge \neg\varphi_n$ is decidable in time $O(n^3)$.

We recall that all our results are worked out for two cases, the structure of finite feature trees and the structure of possibly infinite feature trees.

We reduce the satisfiability problem of Dörre's weak-subsumption constraints [19] over feature algebras in linear time to the one in $\mathrm{FT}_{\leq}$. Thereby, our algorithm improves on the best known satisfiability test for weak subsumption constraints which uses quite different techniques based on finite automata and has an $O(n^5)$-complexity bound [19].

Plan of the Article. Section 2 surveys related work. Section 3 defines the syntax and semantics of constraint system $\mathrm{FT}_{\leq}$ of ordering constraints over feature trees. Section 4 presents a closure algorithm deciding the satisfiability problem of $\mathrm{FT}_{\leq}$. In Section 5, we show how to test entailment and prove the independence property for $\mathrm{FT}_{\leq}$. Section 6 shows that $\mathrm{FT}_{\leq}$ is strictly more expressive than FT. Section 7 defines weak subsumption constraints and reduces their satisfiability problem to the one of $\mathrm{FT}_{\leq}$. Section 8 explains how to implement the closure algorithm for testing satisfiablity in cubic time. Section 9

completes the correctness proofs for the presented satisfiability and entailment tests and Section 10 concludes.

2. Related Work

Ines Constraints. In previous work [32], the authors have introduced the constraint system INES, whose constraints are inclusions between first-order terms interpreted over nonempty sets of trees; the satisfiability test for INES constraints is cubic. The satisfiability test for $\mathrm{FT}_{\leq}$ is inspired by the one for INES. The entailment problems of $\mathrm{FT}_{\leq}$ and INES are different. Intuitively, the entailment problem of $\mathrm{FT}_{\leq}$ is less difficult than the one of INES because a constraint of $FT_{\leq}$ cannot uniquely describe a single feature tree; in contrast, an INES constraint can uniquely describe a constructor tree (i.e. ground term) as a singleton set. For instance, the INES constraint $x{\subseteq}a$ describes the singleton $\{a\}$. As a consequence, the implication $x{\subseteq}a \rightarrow a{\subseteq}x$ holds in INES. The entailment problem of INES constraints is PSPACE-complete in case of an infinite signature and at least DEXPTIME-hard for a finite signature [36]. Previously, it was already noted that the entailment problem of INES constraints is coNP-hard [30]. The algorithm given in [15] is not a complete test of entailment of INES constraints; the one given in [16] applies to a larger class of constraints for the case of an infinite signature and lies in DEXPTIME.

Feature Constraints. The constraint system CFT [48] extends FT by arity constraints of the form $x\{f_1, \ldots, f_n\}$, saying that the denotation of x has subtrees exactly at the features f_1 through f_n. CFT subsumes Colmerauer's rational tree constraint system RT [17] but provides finer-grained constraints. Complete axiomatizations for FT and CFT in case of an infinite signature have been given in [11] and [9], respectively. Due to complete axiomatisation, the first-order theory of FT is decidable.

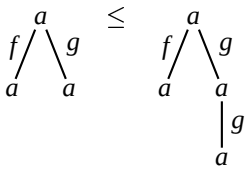
The investigation of ordering constraints over feature trees presented in this paper is continued in two follow-up papers. In [31] it is shown how to express constraints of $FT_{\leq}$ in second-order monadic logic (S2S or WS2S). Thereby, an algorithm for solving the entailment problem of $\mathrm{FT}_{\leq}$ with existential quantifiers ("$FT_{\leq} \models \varphi \rightarrow \exists \bar{x} \varphi'$") was obtained for a first time. Later on, it turned out [33] that the entailment problem of $\mathrm{FT}_{\leq}$ with existential quantifiers is PSPACE-complete (for finite or infinite signatures, and for finite or possibly infinite trees). It was also proved in [33] that the first-order theory of $FT_{\leq}$ is undecidable (in contrast to the first-order theory of FT). The system $\mathrm{FT}_{\leq}$(sort) extends $FT_{\leq}$ by allowing a partial order on labels [29].

The system EF [49] extends CFT by feature constraints $x[y]z$, providing for first-class features. The satisfiability problem of EF constraints is shown NP-complete. Another extension of FT is the system RFT which features so-called regular path expressions [8, 10]

(Weak) Subsumption Constraints. The subsumption and the weak subsumption orderings can be defined for arbitrary feature algebras [19]. In particular, the structure FT of feature trees is a feature algebra (called the algebra of path functions in [19]). As already proved there, the information ordering on feature trees coincides with the weak subsumption ordering of the feature algebra of feature trees.

The subsumption ordering [21] is a subrelation of the weak subsumption ordering. The

converse is not true. For instance, the weak subsumption ordering does hold in the example given in the picture, whereas the subsumption ordering does not. This is because the two equal subtrees of the smaller tree – its leaves – are extended in different manners when moving from the left to the right. The definition of subsumption, however, requires that equal subtrees are extended in the same manner.



Subsumption constraints have been considered in the context of unification-based grammars to model coordination phenomena in natural language [21, 19, 44]. There, one wants to express that two feature structures representing different parts of speech share common properties. For example, the analysis of "programming" and "linguistics" in the phrase

"Feature constraints are good for [$_{NP}$ programming] and [$_{NP}$ linguistics]"

should share (but might refine differently) the information common to all noun phrases. Since the satisfiability of subsumption constraints is undecidable [21], Dörre proposed weak subsumption constraints as a decidable approximation of subsumption constraints.

Independence. A constraint system has the fundamental *independence property* if negated conjuncts are independent from each other. This means that $\varphi \wedge \neg\varphi_1 \wedge \ldots \wedge \neg\varphi_n$ is satisfiable if and only if there exists $1 \leq i \leq n$ such that $\varphi \wedge \neg\varphi_i$ is satisfiable. This is equivalent to that $\varphi \models \varphi_1 \vee \ldots \vee \neg\varphi_n$ holds if and only if there exists $1 \leq i \leq n$ such that $\varphi \models \neg\varphi_i$. The independence property is important since it allows us to use an entailment test for solving negative constraints.

The constrain systems RT, FT, CFT have the independence property in case of an infinite signature [17, 6, 4, 48]. Apart from these, constraint systems with the independence property include linear equations over the real numbers [27], or infinite boolean algebras with positive constraints [22], and set constraints with intersections interpreted over nonempty sets of trees [32, 16, 36].

3. Syntax and Semantics

In this section, we introduce the syntax and semantics of ordering constraints over feature trees. We introduce two systems of ordering constraints – $\mathrm{FT}_{\leq}$ and $FT^{fin}_{\leq}$ – depending on whether we interpret over finite feature trees or over possibly infinite feature trees.

We assume an infinite set of *variables* ranged over by x, y, z, an infinite set $\mathcal{F}$ of *features* ranged over by f, g and an arbitrary set $\mathcal{L}$ of labels denoted by a, b containing at least two distinct elements. The existence of infinitely many features is fundamental for independence and for our entailment algorithm in Section 5 to work. It is irrelevant for the satisfiability test in Section 4.

Feature Trees. A *path* p is a finite sequence of features in $\mathcal{F}$. The *empty path* is denoted by ε and the free-monoid concatenation of paths p and p' as pp'; we have $\varepsilon p = p\varepsilon = p$. Given paths p and q, p' is called a *prefix of* p if $p = p'p''$ for some path p''. A *tree domain* is a non-empty prefixed-closed set of paths.

A *feature tree* τ is a pair (D, L) consisting of a tree domain D and a partial labeling function $L : D \rightharpoonup \mathcal{L}$. Given a feature tree τ, we write D_τ for its tree domain and L_τ for its labeling function. A feature tree is called *finite* if its tree domain is finite, and *infinite* otherwise. Slightly overloading notation, we denote the set of all feature trees by $FT_\leq$ and the set of all finite feature trees with $FT_\leq^{fin}$. If $p \in D_\tau$ we write as $\tau[p]$ the subtree of τ at path p which is formally defined by $D_{\tau[p]} = \{p' \mid pp' \in D_\tau\}$ and $L_{\tau[p]} = \{(p', a) \mid (pp', a) \in L_\tau\}$.

Syntax. An *ordering constraint* φ is defined by the following abstract syntax.

$$\varphi \quad ::= \quad x{\leq}y \mid a(x) \mid x[f]y \mid x{\sim}y \mid \varphi_1 \wedge \varphi_2$$

An ordering constraint is a conjunction of *atomic constraints* which are either *atomic ordering constraints* $x{\leq}y$, *labeling constraints* $a(x)$, *selection constraints* $x[f]y$, or *compatibility constraints* $x{\sim}y$. Compatibility constraints are needed in our algorithm and can be expressed by first-order formulae over ordering constraints (see Proposition 1). We identify ordering constraints up to associativity and commutativity of conjunction, *i.e.*, we view an ordering constraint as a multiset of atomic ordering, labeling, selection, and compatibility constraints. We write φ *in* φ' if all conjuncts in φ are contained in φ'. The *size of a constraint* φ is defined as the number occurrences of features, node labels, and variables in φ.

Semantics. We next define the structures $FT_\leq$ and $FT_\leq^{fin}$ of feature trees and finite feature trees respectively. Throughout the paper, we distinguish two cases depending on whether we interpret ordering constraints of $FT_\leq$ or $FT_\leq^{fin}$. The signatures of both structures contain the binary relation symbols $\leq$ and $\sim$, for every label a a unary relation symbol $a(\,.\,)$, and for every feature f a binary relation symbol $\,.\,[f]\,.\,$. The domain of the structure $FT_\leq$ is the set of possibly infinite feature trees (also called $FT_\leq$), and the domain of the structure $FT_\leq^{fin}$ is the set of finite feature trees (also called $FT_\leq^{fin}$). The relation symbols are interpreted as follows:

$$\begin{array}{rll}
\tau_1{\leq}\tau_2 & \text{iff} & D_{\tau_1} \subseteq D_{\tau_2} \text{ and } L_{\tau_1} \subseteq L_{\tau_2} \\
\tau_1[f]\tau_2 & \text{iff} & D_{\tau_2} = \{p \mid fp \in D_{\tau_1}\} \text{ and } L_{\tau_2} = \{(p, b) \mid (fp, b) \in L_{\tau_1}\} \\
a(\tau) & \text{iff} & (\varepsilon, a) \in L_\tau \\
\tau_1{\sim}\tau_2 & \text{iff} & L_{\tau_1} \cup L_{\tau_2} \text{ is a partial function (on } D_{\tau_1} \cup D_{\tau_2})
\end{array}$$

Notice that the relation $\sim$ is not transitive! For instance, let τ_a be a tree whose root is labeled with a, and τ_b a tree whose root is labeled with b, and $\bullet$ the least tree consisting of a single unlabeled node. If $a \neq b$ then it holds that $\tau_a{\sim}\bullet$ and $\bullet{\sim}\tau_b$ but not $\tau_a{\sim}\tau_b$.

Let Φ denote a first-order formula built from ordering constraints with the usual first order connectives, i.e.: $\Phi ::= \varphi \mid \mathsf{true} \mid \mathsf{false} \mid \neg\Phi \mid \Phi \rightarrow \Phi' \mid \forall x\, \Phi \mid \exists x\, \Phi \mid \Phi \wedge \Phi' \mid \Phi \vee \Phi'$. We denote with $V(\Phi)$, $L(\Phi)$, and $F(\Phi)$, respectively, the set of variables occurring free in Φ, and the set of labels and features occurring in Φ.

Suppose that $\mathcal{A}$ is a structure with the same signature than $FT_\leq^{fin}$ and $FT_\leq$. A *solution of* Φ in $\mathcal{A}$ is a variable assignment α into the domain of $\mathcal{A}$ such that Φ evaluates to true under $\mathcal{A}$ and α. We call Φ *satisfiable in* $\mathcal{A}$ if there exists a solution for Φ in $\mathcal{A}$. A formula Φ is

valid in $\mathcal{A}$ if all variable assignments into the domain of $\mathcal{A}$ are solutions of Φ. We say that $\mathcal{A}$ is a *model* of a set of formulas if all its formulas are valid in $\mathcal{A}$. A constraint Φ *entails* Φ' in $\mathcal{A}$, written $\Phi \models_{\mathcal{A}} \Phi'$ if $\Phi \to \Phi'$ is valid in $\mathcal{A}$; Φ is *equivalent* to Φ' in $\mathcal{A}$ if $\Phi \leftrightarrow \Phi'$ is valid in $\mathcal{A}$.

PROPOSITION 1 *The formulae* $x{\sim}y$ *and* $\exists z(x{\leq}z \wedge y{\leq}z)$ *are equivalent in* $FT_{\leq}$ *and* $FT^{fin}_{\leq}$.

Proof: If is sufficient to prove the proposition for $FT_{\leq}$. Let σ be a variable assignment into $FT_{\leq}$ which solves the formula $\exists z(x{\leq}z \wedge y{\leq}z)$. Since $L_{\sigma(x)} \cup L_{\sigma(y)} \subseteq L_{\sigma(z)}$ and $L_{\sigma(z)}$ is a partial function, $L_{\sigma(x)} \cup L_{\sigma(y)}$ is also a partial function. Hence σ is a solution of $x{\sim}y$. Conversely, if σ is a solution of $x{\sim}y$ then $L_{\sigma(x)} \cup L_{\sigma(y)}$ is a partial function. Thus, the pair $\tau =_{def} (D_{\sigma(x)} \cup D_{\sigma(y)}, L_{\sigma(x)} \cup L_{\sigma(y)})$ is a feature tree and every variable assignment σ' with $\sigma'(z) = \tau$, $\sigma'(x) = \sigma(x)$, and $\sigma'(y) = \sigma(y)$ is a solution of $x{\leq}z \wedge y{\leq}z$. ■

4. Satisfiability Test

We present a set of axioms schemes valid for $FT_{\leq}$ and an extended scheme for $FT^{fin}_{\leq}$ which provides for an additional occurs check. We can interpret both axiom schemes as algorithms which solve the satisfiability problems of $\mathrm{FT}_{\leq}$and $FT^{fin}_{\leq}$ respectively. Note that the axiom schemes given here are inspired by those presented for INES constraints in [32].

Table 1 contains the axiom schemes F1 - F5 for $FT_{\leq}$ and the schemes F1-F6 for $FT^{fin}_{\leq}$. For instance, the scheme $x{\leq}x$ represents the infinite set of axioms obtained by choosing some variable for x. All axioms are of one of the following forms: φ, $\varphi \to \varphi'$, or $\varphi \to \mathsf{false}$. The last two forms are distinct since false is not a constraint.

Schemes F1.1 and F1.2 express the reflexivity and transitivity of the information ordering; F2 says that it has the decomposition property. F3.1 states the reflexivity of the compatibility relation. F3.2 says that if x has less information than y and the information of y and z is compatible, then the information of x and z is also compatible. It follows from the transitivity of the information ordering that $FT_{\leq}$ and $FT^{fin}_{\leq}$ are models of the axioms in F2 (see Proposition 1). F3.3 states the symmetry of the compatibility relation. F4 expresses that the compatibility relation has the decomposition property. Axiom scheme F5 states that two trees cannot be compatible if they carry distinct labels at the root. The last scheme F6 is a version of the occurs check which holds for $FT^{fin}_{\leq}$ but not for $FT_{\leq}$.

PROPOSITION 2 *The structure* $FT_{\leq}$ *is a model of the axioms in* $\mathsf{F1}-\mathsf{F5}$ *and the structure* $FT^{fin}_{\leq}$ *a model of the axioms in* $\mathsf{F1}-\mathsf{F6}$.

Proof: By a routine check. Since it is the most interesting one, we prove the statement for the scheme F3.2, i.e. we show that the formula $x{\leq}y \wedge y{\sim}z \to x{\sim}z$ is valid in $FT_{\leq}$ for all x,y,z. The following implications are valid in $FT_{\leq}$:

$$
\begin{array}{lcll}
x{\leq}y \wedge y{\sim}z & \leftrightarrow & x{\leq}y \wedge \exists u\,(y{\leq}u \wedge z{\leq}u) & \text{Proposition 1} \\
 & \to & \exists u\,(x{\leq}u \wedge z{\leq}u) & \text{Transitivity} \\
 & \leftrightarrow & x{\sim}z & \text{Proposition 1}
\end{array}
$$

■

F1.1	$x \leq x$	
F1.2	$x \leq y \wedge y \leq z \rightarrow x \leq z$	
F2	$x[f]x' \wedge x \leq y \wedge y[f]y' \rightarrow x' \leq y'$	
F3.1	$x \sim x$	
F3.2	$x \leq y \wedge y \sim z \rightarrow x \sim z$	
F3.3	$x \sim y \rightarrow y \sim x$	
F4	$x[f]x' \wedge x \sim y \wedge y[f]y' \rightarrow x' \sim y'$	
F5	$a(x) \wedge x \sim y \wedge b(y) \rightarrow \mathsf{false}$	for $a \neq b$
F6	$\bigwedge_{i=1}^{n} x_i[f_i]y_{i+1} \wedge x_{i+1} \leq y_{i+1} \rightarrow \mathsf{false}$	for $x_{n+1} = x_1$ $\quad (n \geq 1)$

Table 1. Axioms of Satisfiability: F1-F5 for $\mathrm{FT}_{\leq}$ and F1-F6 for $FT^{fin}_{\leq}$

We next present a sequence of examples to show the consequences which can be derived with the given axioms schemes.

EXAMPLE 1 *It is most important that the following axiom scheme can be derived from the schemes* F3.1, F3.2, *and* F3.3*:*

$$x \leq z \wedge y \leq z \rightarrow x \sim y$$

From $x \leq y \wedge x \leq z$, we can derive $z \sim z$ with F3.1 *and thus $x \sim z$ by* F3.2*, then $z \sim x$ via* F3.3*. Another application of* F3.2 *yields $y \sim x$ such that $x \sim y$ follows form* F3.3*.*

EXAMPLE 2 *An inconsistency can be raised by two incompatible* lower *bounds. For instance, consider:*

$$a(x) \wedge x \leq z \wedge y \leq z \wedge b(y) \rightarrow \mathsf{false} \qquad \textit{for } a \neq b$$

As shown in the previous example, we derive $x \sim y$ from $x \leq z \wedge y \leq z$ by using F3.1, F3.2, *and* F3.3*. Hence,* false *can be derived with* F5.

In contrast to lower bounds, *upper* bounds are always compatible. For instance, the analoguous constraint to above, $a(x) \wedge z \leq x \wedge z \leq y \wedge b(y)$, is satisfiable since z can be chosen to denote the least tree consisting of a single unlabeled node.

EXAMPLE 3 *The rule* F4 *is perhaps the key rule for deriving inconstencies. This can be illustrated as follows:*

$$a(x) \wedge x[g]x \wedge x{\leq}z \wedge y{\leq}z \wedge y[g]y' \wedge b(y') \rightarrow \mathsf{false} \qquad \textit{for } a \neq b$$

As shown in Example 1, we can derive $x{\sim}y$ *from* $x{\leq}z \wedge y{\leq}z$ *by using* F3.1, F3.2, *and* F3.3. *We can now apply* F4 *to* $x[g]x \wedge x{\sim}y \wedge y[g]y'$ *in order to derive* $x{\sim}y'$. *Finally, this allows us to derive* false *from* $a(x) \wedge x{\sim}y' \wedge b(y')$ *via* F5.

EXAMPLE 4 *The constraint* $x[f]y \wedge x{\leq}y$ *is unsatisfiable in* $FT_{\leq}^{fin}$ *but satisfiable in* $FT_{\leq}$. *In the first case, we can apply the occurs check* F6 *in order to derive* false. *In the second case, the occurs check is not valid.*

The Algorithm F. In case of $FT_{\leq}$, we define F to be the set of axiom schemes F1-F5, whereas in case of $FT_{\leq}^{fin}$, we define F to be the set of schemes $\mathsf{F1} - \mathsf{F6}$. Both sets induce a closure algorithm that we also call F. These algorithms input a constraint φ and add iteratively new logical consequences of $\mathsf{F} \cup \{\varphi\}$ to φ.

More precisely, every step of F inputs a constraint φ and then terminates with false, or terminates with φ, or passes over a constraint of the form $\varphi \wedge \varphi'$ to the next step. Termination with false occurs if there exists φ'' *in* φ such that $\varphi'' \rightarrow \mathsf{false}$ is an instance of an axiom scheme in F. Termination with φ happens if no new constraint can be added to φ. Recursion with $\varphi \wedge \varphi'$ is possible if φ' is an instance of an axiom scheme in F which satisfies $\mathcal{V}(\varphi') \subseteq \mathcal{V}(\varphi)$, or if there there exists φ'' *in* φ for which $\varphi'' \rightarrow \varphi'$ is an instance of an axiom scheme in F.

If G is a subset of the set of axiom schemes F then we call a constraint φ *G-closed* if no new consequence can be added to φ by applying an axiom scheme in G (in the way defined above). We note that false is not a constraint and hence cannot be F-closed.

PROPOSITION 3 *If* φ *is a constraint of size* m *then the algorithm* F *started with input* φ *terminates in at most* $2 \cdot m^2$ *steps (where* F1.1 *and* F3.1 *are applied to variables in* φ *only).*

Proof: Since F does not introduce new variables, it may add at most m^2 new compatibility constraints $x{\sim}y$ and m^2 new atomic ordering constraints $x{\leq}y$. With respect to not adding new variables, only the scheme F1.1 and F3.3 are critical. Both of these are of the form φ such that their application cannot introduce new variables by definition. ■

EXAMPLE 5 *Algorithm* F *terminates in presence of cyclic constraints like* $x[f]x$. *For instance, the following constraint is* $\mathsf{F1} - \mathsf{F5}$*-closed but not* F6*-closed.*

$$x[f]x \wedge x{\leq}y \wedge y[f]y \wedge x{\leq}x \wedge y{\leq}y \wedge x{\sim}x \wedge y{\sim}y \wedge x{\sim}y \wedge y{\sim}x$$

In particular, F2 *and* F4 *do not loop through the cycle* $x[f]x$. *This example illustrates the need of compatibility constraints. Without them one might wish to apply the following rule whereby a new variable* z *is introduced which raises non-termination:*

$$x[f]x' \wedge x{\leq}y \rightarrow \exists z(y[f]z \wedge x'{\leq}z)$$

Remark. Notice that Table 1 deliberately leaves out the following scheme that is perfectly valid in $FT_{\leq}$ and $FT_{\leq}^{fin}$. This is done for simplicity only; it would do no harm adding.

$$x \leq y \wedge a(x) \rightarrow a(y)$$

Definition 1. Let φ be a constraint. If algorithm F started with φ terminates and returns a constraint (but not false) then we call its result the F-closure of φ and denote it by $\mathrm{cl}(\varphi)$.

Note that $\mathrm{cl}(\varphi)$ is not defined for all φ but always F-closed when defined. Since the definition of F is parametrized by the choice of $FT_{\leq}$ or $FT^{fin}_{\leq}$, the definition of $\mathrm{cl}(\varphi)$ is also parametrized by one of these structures. It is not possible, however, that $\mathrm{cl}(\varphi)$ differs for $FT_{\leq}$ and $FT^{fin}_{\leq}$. In the worst case, $\mathrm{cl}(\varphi)$ exists with respect to $FT_{\leq}$ but not for $FT^{fin}_{\leq}$. This happens if false can be derived by applying $\mathsf{F6}$ to the F-closure of φ with respect to $FT_{\leq}$.

PROPOSITION 4 *Every* $\mathsf{F1}-\mathsf{F5}$*-closed constraint is satisfiable in* $FT_{\leq}$ *and every* $\mathsf{F1}-\mathsf{F6}$*-closed constraint is satisfiable in* $FT^{fin}_{\leq}$.

Proof: See Section 9.2. ■

THEOREM 1 *The satisfiability problem of* $FT_{\leq}$ *and* $FT^{fin}_{\leq}$ *can be decided (off-line and on-line) in cubic time in size of the input constraint. The* F*-closure of a satisfiable constraint exists and can be computed in cubic time.*

Proof: Proposition 2 shows that φ is unsatisfiable if algorithm F started with φ terminates with false. Proposition 4 proves that φ is satisfiable if F started with φ terminates with an F-closed constraint. Since F terminates for all input constraints (Proposition 3) this yields an effective decision procedure for testing satisfiability and computing the F-closure of a satisfiable constraint. The main idea of the complexity proof is that one needs at most $O(n^2)$ steps where n is the size of the input constraint (Proposition 3) each of which can be implemented in time $O(n)$. The implementation can be organized incrementally by exploiting the fact that algorithm F leaves unspecified the order in which the axioms are applied. Hence, we obtain that off-line and on-line complexity are the same. The implementation is detailed in Section 8 ■

5. Entailment, Independence, Negation

In this section, we give a cubic time algorithm for testing entailment of ordering constraints over feature trees. Our algorithm is parametrized by a structure – either $FT_{\leq}$ or $FT^{fin}_{\leq}$ – and, depending on the particular parameter, decides entailment judgments of the form $\varphi \models_{FT_{\leq}} \varphi'$ or $\varphi \models_{FT^{fin}_{\leq}} \varphi'$. The structure chosen is relevant only for a single subroutine of the entailment test, which is the satisfiability test presented in the previous section.

We also prove the independence property for the constraint languages $\mathrm{FT}_{\leq}$ and $FT^{fin}_{\leq}$. Based on the independence property, we show how to solve conjunctions of positive and negative ordering constraints $\varphi \wedge \neg\varphi_1 \wedge \ldots \neg\varphi_n$ in time $O(n^3)$. Note that all results of this section depend on the existence of infinitely many features in the given signature.

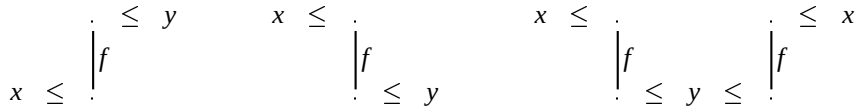


Table 2. The respective graphs of a constraint which supports $x{\leq}y[f]$, $x?[f]{\leq}y$, or $x[f]y$ syntactically.

For the rest of this section, we fix either of the structures $FT_{\leq}$ or $FT_{\leq}^{fin}$; we write $\varphi \models \varphi'$ rather than $\varphi \models_{FT_{\leq}} \varphi'$ or $\varphi \models_{FT_{\leq}^{fin}} \varphi'$.

We denote with μ an atomic constraint, i.e. μ is always a conjunction free ordering constraint ($\mu ::= x{\leq}y \mid x{\sim}y \mid a(x) \mid x[f]y$). Note that an entailment judgment $\varphi \models \varphi'$ holds if and only if the entailment judgments $\varphi \models \mu$ hold for all atomic constraints μ *in* φ'. Next we characterize entailment problems $\varphi \models \mu$ syntactically. For atomic constraints μ of the form $x{\leq}y$, $x{\sim}y$, or $a(x)$, we say that a constraint φ *syntactically supports* μ, written $\varphi \vdash \mu$, if one of the following holds:

$$
\begin{array}{lll}
\varphi \vdash a(x) & \text{if} & \text{exists } x' \text{ such that } x'{\leq}x \wedge a(x') \text{ in } \varphi \\
\varphi \vdash x{\leq}y & \text{if} & x{\leq}y \text{ in } \varphi \text{ or } x = y \\
\varphi \vdash x{\sim}y & \text{if} & x{\sim}y \text{ in } \varphi \text{ or } x = y
\end{array}
$$

The definition of syntactic support of selection constraints, $\varphi \vdash x[f]y$, is slightly more involved. For its definition, we make use of two simple forms of auxiliary *path constraints*, $x{\leq}y[f]$ and $x?[f]{\leq}y$. A path constraint of the first form $x{\leq}y[f]$ requires that the tree for y have feature f and that its subtree at f be greater than the tree for x. A path constraint of the second form $x?[f]{\leq}y$ reads as follows: if the tree for x has feature f then its subtree at f is smaller than the tree for y. We next define the notions of syntactic support for path constraints and selection constraints; this definition is illustrated graphically in Table 2.

$$
\begin{array}{lll}
\varphi \vdash x{\leq}y[f] & \text{if} & \text{exist } x', y' \text{ such that } x{\leq}x' \wedge y'[f]x' \wedge y'{\leq}y \text{ in } \varphi \\
\varphi \vdash x?[f]{\leq}y & \text{if} & \text{exist } x', y' \text{ such that } x{\leq}x' \wedge x'[f]y' \wedge y'{\leq}y \text{ in } \varphi \\
\varphi \vdash x[f]y & \text{if} & \varphi \vdash y{\leq}x[f] \text{ and } \varphi \vdash x?[f]{\leq}y
\end{array}
$$

PROPOSITION 5 (CORRECTNESS) *For all* F*-closed constraints* φ *and atomic constraints* μ*:* $\varphi \vdash \mu$ *implies* $\varphi \models \mu$.

Proof: The cases for μ being of the forms $a(x)$, $x{\leq}y$, or $x{\sim}y$ are obvious. Now, we consider the case that μ is a selection constraint, say $x[f]y$. If $\varphi \vdash \mu$ then $\varphi \vdash y{\leq}x[f]$ *and* $\varphi \vdash x?[f]{\leq}y$ hold. Let α be a solution of φ. Because of $\varphi \vdash y{\leq}x[f]$ it holds that $f \in D_{\alpha(x)}$ and $\alpha(y){\leq}\alpha(x)[f]$. The assumption $\varphi \vdash x?[f]{\leq}y$ yields $\alpha(x)[f]{\leq}\alpha(y)$ if $f \in D_{\alpha(x)}$. We already know that $f \in D_{\alpha(x)}$ is valid; thus $\alpha(y){\leq}\alpha(x)[f]{\leq}\alpha(y)$ holds. So far, we have proved $f \in D_{\alpha(x)}$ and $\alpha(y) = \alpha(x)[f]$, i.e. that α is a solution of $x[f]y$. ■

We show next that syntactic support is strong enough to characterize entailment (Proposition 6) and investigate the complexity of deciding syntactic support (Lemma 2). In combination with the cubic satisfiability test of the previous section, we obtain a cubic entailment test (Theorem 3).

The most difficult claim to show is that syntactic support is complete with respect to entailment: That is, that no atomic constraint μ is entailed by a constraint φ if μ is not already supported in φ. To show this we assume that φ does not support μ syntactically and define a solution of φ that contradicts μ. As we show, there even exists a single solution that contradicts all μ built from symbols in φ *at the same time*. We prove this by giving a satisfiable formula that strengthens φ and entails the negation of all relevant μ's. We call such a formula *saturated*.

LEMMA 1 (EXISTENCE OF A SATURATED FORMULA) *For every satisfiable constraint φ, there exists a formula* $\mathrm{Sat}(\varphi)$*, called a* saturation *of φ, with the following properties.*

1. $\mathrm{Sat}(\varphi)$ *is satisfiable.*
2. $\mathrm{Sat}(\varphi) \models \varphi$.
3. *for all μ if $V(\mu) \subseteq V(\varphi)$ and $F(\mu) \subseteq F(\varphi)$ then $\varphi \not\vdash \mu$ implies* $\mathrm{Sat}(\varphi) \models \neg\mu$.

Proof: The proof is postponed to the end of Section 9.3. ■

THEOREM 2 (INDEPENDENCE) *If the set of features is infinite then both languages $FT_{\leq}$ and $FT_{\leq}^{fin}$ of ordering constraints over feature trees have the independence property: For every $n \geq 1$ and constraints $\varphi, \varphi_1, \ldots, \varphi_n$:*

$$\textit{if} \quad \varphi \models \bigvee_{i=1}^{n} \varphi_i \quad \textit{then} \quad \varphi \models \varphi_j \quad \textit{for some } j \in \{1, \ldots, n\}.$$

Proof: Assume $\varphi \models \bigvee_{i=1}^{n} \varphi_i$. If φ is unsatisfiable we are done. Also, if $\varphi \wedge \varphi_j$ is non-satisfiable for some j, then:

$$\varphi \models \bigvee_{i=1}^{n} \varphi_i \quad \text{iff} \quad \varphi \models \bigvee_{i=1, i\neq j}^{n} \varphi_i \,.$$

Hence we can, without loss of generality, assume that φ and $\varphi \wedge \varphi_i$ are satisfiable for all i, and that φ is F-closed. If there exists an i such that $\varphi \vdash \mu$ for all atomic constraints μ *in* φ_i, then $\varphi \models \varphi_i$ by correctness of syntactic support (Proposition 5) and we are done. Otherwise, for all i there exists μ_i *in* φ_i such that $\varphi \not\vdash \mu_i$. Let $\mathrm{Sat}(\varphi)$ be the formula postulated by Lemma 1. Without loss of generality, we can assume that $V(\varphi_i) \subseteq V(\varphi)$ for all i. Hence $V(\mu_i) \subseteq V(\varphi_i)$ implies $\mathrm{Sat}(\varphi) \models \neg\mu_i$ by Property 3, such that:

$$\mathrm{Sat}(\varphi) \models \bigwedge_{i=1}^{n} \neg\varphi_i \,.$$

Since $\mathrm{Sat}(\varphi)$ is satisfiable and entails φ (Properties 1 and 2), this contradicts our assumption that $\varphi \models \bigvee_{i=1}^{n} \varphi_i$. ■

EXAMPLE 6 *Independence fails in case of a finite set of features. For illustration, assume $\mathcal{F} = \{f, g\}$. If the set of node labels is finite, say $\mathcal{L} = \{a, b\}$ then following entailment judgments holds:*

$$x[f]z \wedge x[g]z \wedge y[f]z \wedge y[g]z \quad \models \quad a(x) \vee b(x) \vee x{\leq}y$$

But neither of the disjunctions on the right hand side is entailed by the left hand side.

EXAMPLE 7 *For a finite set of features and an infinite set of node labels, we can still construct a counter example for independence in case of $FT_{\leq}$ which does however not apply to $FT_{\leq}^{fin}$. In fact, the following counter example applies to all signatures with $\mathcal{F} = \{f, g\}$ and $\mathcal{L} \neq \emptyset$:*

$$\begin{array}{lcl} x[f]x \wedge x[g]x & \wedge & y[f]y \wedge y[g]y \\ z[f]z \wedge z[g]z & \wedge & x{\leq}y \wedge a(y) \end{array} \quad \models \quad y{\leq}x \vee x{\leq}z$$

To see this, notice that $x[f]x \wedge x[g]x$ implies that the tree for x is homogeneously labeled (i.e., either completely unlabeled or labeled with the same symbol at all nodes). The same holds for the trees for y and z. If the tree for x is completely unlabeled then $x{\leq}z$ follows. Otherwise, the tree for x must be labeled with a at all nodes due to $x{\leq}y \wedge a(y)$ such that the trees for x and y are equal: hence $y{\leq}x$ follows.

Independence of $FT_{\leq}$ or $FT_{\leq}^{fin}$ depends strongly on the fact that these constraint languages do not provide for existential quantification. This is illustrated by the following example. If, say $\mathcal{L} = \{a, b\}$, then every feature tree is labeled with a or with b unless it is unlabeled. Therefore, the following entailment judgment holds for all φ.

$$\varphi \models a(x) \vee b(x) \vee \mathsf{unlabeled}(x)$$

Or course, none of the conjuncts of the right hand side is entailed when choosing the left hand side φ to be $x{\leq}x$. Furthermore, the formula $\mathsf{unlabeled}(x)$ can be expressed with existential quantification:

$$\mathsf{unlabeled}(x) \leftrightarrow \exists y \exists z\,(x{\sim}y \wedge a(y) \wedge x{\sim}z \wedge b(z))$$

Hence, a language of ordering constraints over feature trees extended by existential quantification does *not* have the independence property. This failure can be interpreted as a first hint to that entailment which existential quantification is much harder to decide than without. In fact, the entailment problems of $FT_{\leq}$ and $FT_{\leq}^{fin}$ with existential quantification are both PSPACE-complete as shown in [33].

PROPOSITION 6 (CHARACTERIZATION) *The notions of entailment and of syntactic support coincide for atomic constraints, in the sense that if φ is F-closed and μ an atomic constraint then $\varphi \models \mu$ iff $\varphi \vdash \mu$.*

Proof: Syntactic support is *semantically correct* by Proposition 5. It remains to show that syntactic support is *semantically complete*, *i.e.*, $\varphi \models \mu$ implies $\varphi \vdash \mu$. So, assume $\varphi \models \mu$. If $V(\mu) \not\subseteq V(\varphi)$ then μ is of the form $x{\leq}x$ or $x{\sim}x$ such that $\varphi \vdash \mu$ is trivial. Otherwise, assume $V(\mu) \subseteq V(\varphi)$. Now let $\mathrm{Sat}(\varphi)$ be the saturation formula postulated by Lemma 1. By Property 2, $\varphi \models \mu$ implies $\mathrm{Sat}(\varphi) \models \mu$. With Property 1, this yields $\mathrm{Sat}(\varphi) \not\models \neg\mu$, and Property 3 implies $\varphi \vdash \mu$. ■

LEMMA 2 *Given an* F*-closed constraint* φ *of size* n*, we can compute a representation of* φ *in time* $O(n^2)$ *that allows for testing syntactic support* $\varphi \vdash \mu$ *in time* $O(n^2)$.

Proof: As a representation for the F-closed constraint φ, we can use 4 arrays of size $O(n^2)$, each of which gives access to one form of atomic constraints, by indexing over variables and features (for details see Section 8). These arrays can be allocated in time $O(n^2)$ and support a test of membership to φ for an atomic constraint in time $O(1)$. Hence, we can check syntactic support for atomic ordering and compatibility constraints in time $O(1)$. For testing $\varphi \vdash a(x)$, we have to find all x' with $x{\leq}x'$ *in* φ and then to test whether one x' satisfies $a(x')$ *in* φ; this can be done in time $O(n)$.
For checking $\varphi \vdash x{\leq}y[f]$, we first compute in time $O(n)$ the set of all x' such that $x{\leq}x'$ *in* φ. From this set, we deduce in time $O(n^2)$ the set of all y' such that $y'[f]x'$ *in* φ for some x' computed above. Finally, we check in time $O(n)$ whether $y'{\leq}y$ *in* φ for at least one y'. The procedures for testing $\varphi \vdash x?[f]{\leq}y$ and $\varphi \vdash x[f]y$ can be organized in analogy. ■

THEOREM 3 (ENTAILMENT) *If the set of features is infinite, then entailment judgments of the form* $\varphi \models_{FT_{\leq}} \varphi'$ *and* $\varphi \models_{FT_{\leq}^{fin}} \varphi'$ *can be tested in cubic time in the size of* $\varphi \wedge \varphi'$.

Proof: Let n be the size of $\varphi \wedge \varphi'$. To decide $\varphi \models \varphi'$, we first test whether or not φ is satisfiable, and return its F-closure $\mathrm{cl}(\varphi)$ in case of satisfiability. By Theorem 1 this can be done in time $O(n^3)$. If φ is not satisfiable then entailment holds trivially. Otherwise, it suffices to test whether $\mathrm{cl}(\varphi) \models \varphi'$ holds. According to Proposition 6 this is equivalent to that $\mathrm{cl}(\varphi) \vdash \mu$ holds for all μ *in* φ'. Since there are $O(n)$ such μ each of which can be tested in time $O(n^2)$ by Lemma 2, syntactic support for all μ *in* φ' is decidable in time $O(n^3)$. Hence, the overall time for testing entailment is also $O(n^3)$. ■

COROLLARY 1 (NEGATION) *The satisfiability in* $FT_{\leq}$ *or* $FT_{\leq}^{fin}$ *of conjunctions of positive and negative ordering constraints of the form* $\varphi \wedge \neg\varphi_1 \wedge \ldots \wedge \neg\varphi_k$ *can be tested in time* $O(n^3)$ *where* n *is the size of the considered formula.*

Proof: If φ is non-satisfiable then $\varphi \wedge (\wedge_{i=1}^{k} \neg\varphi_i)$ is trivially non-satisfiable. By Proposition 1, satisfiability of φ is decidable in time $O(n^3)$ where n is the size of φ. Now assume φ to be satisfiable. By Theorem 2 on independence, $\varphi \wedge (\wedge_{i=1}^{k} \neg\varphi_i)$ is non-satisfiable if and only if $\varphi \models \varphi_i$ for some $1 \leq i \leq k$. This is equivalent to saying that, for some i, μ *in* φ_i implies $\varphi \models \mu$. By Proposition 6 it thus suffices to test $\varphi \vdash \mu$ for all μ *in* φ_i and all $1 \leq i \leq k$. Overall, there are $O(n)$ such μ's to be tested for syntactic support. By Lemma 2, $\varphi \vdash \mu$ can be tested in time $O(n^2)$ such that the total complexity sums up to time $O(n^3)$. ■

6. Expressiveness

We show that $FT_{\leq}$ is strictly more expressive than *FT* [6, 11] but does not generalize CFT [48, 12] in that it cannot express conjunctions of arity and ordering constraints.

The constraint system *FT* is the language of equality constraints interpreted in the structure of completely labeled feature trees. A constraint η of *FT* has the following form:

$$\eta ::= x{=}y \mid a(x) \mid x[a]y \mid \eta \wedge \eta'$$

As for ordering constraints, we can distinguish two cases: In FT^{fin}, we interpret over finite feature trees and in *FT* over possibly infinite ones. For simplicity, we only consider *FT* in this section; our results, however, hold for FT^{fin} in analogy.

The constraint system *CFT* extends *FT* by arity constraints. An *arity constraint* has the form $x\{f_1,\ldots,f_n\}$ and holds if x denotes a tree which has direct subtrees exactly at the features f_1 through f_n.

In the light of the complete axiomatization of the first-order theories of *FT* and *CFT* [11, 12] (which apply for infinite sets of features and labels) we freely permit ourselves to interpret constraints of *FT* over partially labeled feature trees (rather than over completely labeled ones). When doing so, every constraint of *FT* can trivially be expressed in $FT_{\leq}$.

In order to be precise, we define what it means for a formula to *express* a predicate on feature trees. Our definition is well known in mathematical logics and was investigated for feature logics by Backofen [7]: An *n-ary predicate* $\mathcal{P}$ is an n-ary relation between feature trees. We write $\mathcal{P}(\tau_1,\ldots,\tau_n)$ if $(\tau_1,\ldots,\tau_n) \in \mathcal{P}$. We denote a formula Φ with free variables $x_1,\ldots,x_n$ by $\Phi(x_1,\ldots,x_n)$ whereby an ordering on the variables of Φ is fixed.

Definition 2. An n-ary predicate $\mathcal{P}$ *is expressed by a formula* $\Phi(x_1,\ldots,x_n)$ with free variables $x_1,\ldots,x_n$ if for all feature trees $\tau_1,\ldots,\tau_n$:

$$\mathcal{P}(\tau_1,\ldots,\tau_n) \text{ holds iff } \begin{cases} \text{there exists a solution } \alpha \text{ of } \Phi(x_1,\ldots,x_n) \\ \text{such that } \alpha(x_1)=\tau_1,\ldots,\alpha(x_n)=\tau_n \end{cases}$$

PROPOSITION 7 *There is no constraint in* $FT_{\leq}$ *which expresses the fact that a variable* x *denotes the least feature tree* $\bullet$*, i.e., if* $a \neq b$ *then there is no constraint equivalent to:*

$$x\{\} \wedge x{\sim}y \wedge x{\sim}z \wedge a(y) \wedge b(z)$$

Proof: If φ were such an ordering constraint of $FT_{\leq}$ then φ as well as its F-closure would entail $x{\leq}y$ for all variables y. This contradicts Proposition 6 for all those y with $y \notin V(\varphi)$ and $x \neq y$ (because if $\varphi \vdash x{\leq}y$ then $x = y$ or $x,y \in V(\varphi)$). Trivially, a variable $y \notin V(\varphi)$ exists since $V(\varphi)$ is finite. ■

LEMMA 3 *If* η *is a constraint of FT then* $\eta \models x{\leq}y$ *holds iff* $\eta \models y{\leq}x$ *is valid.*

Proof: Let η be a constraint of *FT* and let φ be the ordering constraint obtained from η by replacing all equalities $x{=}y$ in η by an ordering constraint $x{\leq}y \wedge y{\leq}x$. Hence, for

all x,y it holds that $x{\leq}y$ *in* φ iff $y{\leq}x$ *in* φ. Since the closure algorithm F preserves this property of φ, it also holds for $\mathrm{cl}(\varphi)$. Thus, the claim follows from Proposition 6 again. ■

PROPOSITION 8 *If $x \neq y$ then there is no equality constraint of FT equivalent to $x{\leq}y$.*

Proof: This follows immediately from Lemma 3 and Proposition 6. ■

7. Weak Subsumption Constraints

We next compare the constraint language $FT_{\leq}$ to the system of weak subsumption constraints as introduced in [19]. We show that the satisfiability problem of weak subsumption constraints is subsumed by the one for $FT_{\leq}$. Here, interpretation over possibly infinite feature trees is crucial.

Syntax and Semantics. Following [19], a weak subsumption constraint is an ordering constraint φ without compatibility constraints. Since the latter restriction is not crucial we here consider weak subsumption constraints extended with compatibility constraints in order to simplify our comparison.

Weak subsumption constraints are interpreted over the class of all feature algebras, each of which induces a weak subsumption ordering (see below). A *feature algebra* $\mathcal{A}$ with features $\mathcal{F}$ and node labels $\mathcal{L}$ consists of a set $\mathsf{dom}^{\mathcal{A}}$ that is called the *domain* of $\mathcal{A}$, a unary relation $a(.)^{\mathcal{A}}$ on $\mathsf{dom}^{\mathcal{A}}$ for every node label $a \in \mathcal{L}$, and a binary relation $.[f]^{\mathcal{A}}.$ on $\mathsf{dom}^{\mathcal{A}}$ for every feature $f \in \mathcal{F}$[2]. The relations of a feature algebra $\mathcal{A}$ satisfy the following properties for all $\alpha, \alpha', \alpha'' \in \mathsf{dom}^{\mathcal{A}}$, node labels $a, a_1, a_2 \in \mathcal{L}$, and features $f \in \mathcal{F}$:

1. if $\alpha[f]^{\mathcal{A}}\alpha'$ and $\alpha[f]^{\mathcal{A}}\alpha''$ then $\alpha' = \alpha''$
2. if $a_1(\alpha)^{\mathcal{A}}$ and $a_2(\alpha)^{\mathcal{A}}$ then $a_1 = a_2$

In the literature [46, 19] a slightly different notion of feature algebra was considered that we call *feature algebras with constants* here. We will give a formal comparison at the end of the section.

Again overloading notation, let FT be the structure of feature trees with features $\mathcal{F}$ and node labels $\mathcal{L}$, but with a restricted signature in which the relation symbols $\leq$ and $\sim$ are not provided.

PROPOSITION 9 *The structure FT of feature trees is a feature algebra.*

Proof: Property 1 of a feature algebra follows from $\mathsf{F}2$ and property 2 from $F5$. ■

Given a feature algebra $\mathcal{A}$, we define the weak subsumption ordering $\leq^{\mathcal{A}}$ as follows. A *simulation for* $\mathcal{A}$ is a binary relation $\Delta \subseteq \mathsf{dom}^{\mathcal{A}} \times \mathsf{dom}^{\mathcal{A}}$ that satisfies the following properties for all labels a, features f, and all elements α_1, α_2, α'_1, $\alpha'_2 \in \mathsf{dom}^{\mathcal{A}}$:

1. If $\alpha_1 \Delta \alpha_2$, $a(\alpha_1)^{\mathcal{A}}$ then $a(\alpha_2)^{\mathcal{A}}$.

2. If $\alpha_1 \Delta \alpha_2$, $\alpha_1[f]^{\mathcal{A}}\alpha_1'$ then exists $\alpha_2' : \alpha_2[f]^{\mathcal{A}}\alpha_2'$ and $\alpha_1'\Delta\alpha_2'$.

The *weak subsumption ordering* $\leq^{\mathcal{A}}$ of $\mathcal{A}$ is the greatest simulation relation for $\mathcal{A}$. The weak subsumption relation on $\mathcal{A}$ induces a compatibility relation $\sim^{\mathcal{A}}$ through:

$$\alpha_1 \sim^{\mathcal{A}} \alpha_2 \quad \text{iff} \quad \text{exists } \alpha \text{ such that } \alpha_1 \leq^{\mathcal{A}} \alpha \text{ and } \alpha_2 \leq^{\mathcal{A}} \alpha$$

A feature algebra $\mathcal{A}$ induces a structure with the same signature as $FT_{\leq}$ in which $\leq$ is interpreted as the weak subsumption ordering $\leq^{\mathcal{A}}$, $\sim$ as the compatibility relation $.\sim^{\mathcal{A}}.$, $a(.)$ as $a(.)^{\mathcal{A}}$, and $.[f].$ as $.[f]^{\mathcal{A}}.$.

PROPOSITION 10 (DÖRRE [20]) *The structure $FT_{\leq}$ coincides with the structure induced by the feature algebra FT.*

Proof: It is sufficient to prove that the weak subsumption ordering of the feature algebra FT coincides with the information ordering on $FT_{\leq}$. The proof in the case of feature algebras with constants can be found in [20] on page 24 (Satz 6 and Satz 7). There, the structure of feature trees has been called algebra of path functions. We recall the proof for sake of completeness. The information ordering is a simulation for $FT_{\leq}$ due to the axioms in $\mathsf{F1}-\mathsf{F5}$ and hence smaller than the weak subsumption ordering of $FT_{\leq}$. Conversely, we show that every simulation on $FT_{\leq}$ is smaller than its information ordering. Let Δ be a simulation and τ_1, τ_2 feature trees such that $\tau_1 \Delta \tau_2$. We have to show that $\tau_1 \leq \tau_2$. This is equivalent to $D_{\tau_1} \subseteq D_{\tau_2}$ and $L_{\tau_1} \subseteq L_{\tau_2}$ and can be proved by induction on paths. ■

THEOREM 4 *An ordering constraint φ is satisfiable over $FT_{\leq}$ if and only if φ is satisfiable over the structure induced by some feature algebra $\mathcal{A}$.*

Proof: If φ is satisfiable in $FT_{\leq}$ then it is satisfiable in the structure induced by FT (which is $FT_{\leq}$). Conversely, every structure induced by a feature algebra is a model of the axioms in $\mathsf{F1}-\mathsf{F5}$. Thus, if φ is satisfiable in such a structure then it is equivalent to an $\mathsf{F1}-\mathsf{F5}$-closed constraint (and not false) and hence satisfiable over $FT_{\leq}$. ■

Alternative Notions of Feature Algebras. In the literature [46, 19] a restricted notion of feature algebras has been considered that we call *feature algebras with constants* in the sequel. The notion of a feature algebra with constants leads to a restricted satisfiability problem. This shows that the presented results properly extend the results in [19].

A *feature algebra with constants* is a feature algebra with satisfies the following additional property for all labels a and feature f:

$$\text{if } a(\alpha)^{\mathcal{A}} \text{ then not } \alpha[f]^{\mathcal{A}}\alpha'$$

This means that nodes labels behave like constants in terms. In order to handle this new property we consider the following mapping of weak subsumption constraints over $\mathcal{L}$ and $\mathcal{F}$ to weak subsumption constraints over $\mathcal{L}$ and $\mathcal{F} \cup \{\mathsf{label}\}$ where label is a new feature not contained in $\mathcal{F}$.

$$[\![a(x)]\!] = \exists y(x[\mathsf{label}]y \wedge a(y)) \qquad [\![x[f]y]\!] = x[f]y \qquad [\![x{\leq}y]\!] = x{\leq}y$$
$$[\![x{\sim}y]\!] = x{\sim}y \qquad [\![\varphi \wedge \varphi']\!] = [\![\varphi]\!] \wedge [\![\varphi']\!]$$

PROPOSITION 11 *A constraint φ is satisfiable in the structure induced by some feature algebra if and only if $[\![\varphi]\!]$ is satisfiable in the structure induced by some feature algebra with constants.*

Proof: If $[\![\varphi]\!]$ is satisfiable over a feature algebra $\mathcal{A}$ with constants and features $\mathcal{F} \cup \{\mathsf{label}\}$ then φ is satisfiable over the feature algebra $FT_{\leq}$ with features $\mathcal{F}$. Given a solution σ' of $[\![\varphi]\!]$ over $\mathcal{A}$ a solution σ of φ over $FT_{\leq}$ can be defined as follows where we write $[f_1 \ldots f_n]^{\mathcal{A}}$ for $[f_1]^{\mathcal{A}} \circ \ldots \circ [f_n]^{\mathcal{A}}$ if $f_1 \ldots f_n \in \mathcal{F}^*$.

$$\begin{aligned} D_{\sigma(x)} &= \{p \mid \text{ exists } \alpha \text{ in the domain of } \mathcal{A}\text{: } \sigma'(x)[p]^{\mathcal{A}}\alpha \text{ and } p \in \mathcal{F}^*\} \\ L_{\sigma(x)} &= \{(p,a) \mid \text{ exists } \alpha \text{ in the domain of } \mathcal{A} : \sigma'(x)[p\ \mathsf{label}]^{\mathcal{A}}\alpha \text{ and } a(\alpha)^{\mathcal{A}}\} \end{aligned}$$

Conversely, let φ be satisfiable for some feature algebra $\mathcal{A}$. Then φ is satisfiable in $FT_{\leq}$ by Theorem 4. We define a feature algebra with constants FT^{con} and show that $[\![\varphi]\!]$ is satisfiable over FT^{con}. The labels and features of FT^{con} are $\mathcal{L}$ and $\mathcal{F} \cup \{\mathsf{label}\}$, respectively. The domain of FT^{con} contains all feature trees τ without labeled inner nodes, where a *labeled inner node* of τ is a path p such that $p \in D_\tau$, exists a with $(p,a) \in L_\tau$ and exists f with $pf \in D_\tau$. The selection and labeling relations of FT^{con} are those of $FT_{\leq}$ restricted to trees without labeled inner nodes. Obviously, FT^{con} satisfies all three axioms of a feature algebra with constants. Now let σ be a solution of φ in the structure induced by $\mathcal{A}$. Then the variable assignment σ' mapping x on $\sigma'(x)$ as given below is a solution of $[\![\varphi]\!]$ in the structure induced by FT^{con}.

$$\begin{aligned} D_{\sigma'(x)} &= D_{\sigma(x)} \cup \{p\ \mathsf{label} \mid \text{ exists } a \in \mathcal{L} : (p,a) \in L_{\sigma(x)}\} \\ L_{\sigma'(x)} &= \{(p\ \mathsf{label}, a) \mid (p,a) \in L_{\sigma(x)}\} \end{aligned}$$ ■

8. Implementing the Closure Algorithm

We present an implementation of the closure algorithm F for testing satisfiability, thereby proving the complexity statement left open in the proof of Theorem 1. Recall that the algorithm F computes the closure of a constraint whenever it exists with respect to the axioms schemes in Table 1, $\mathsf{F1}-\mathsf{F5}$ in case of $FT_{\leq}$ and $\mathsf{F1}-\mathsf{F6}$ for $FT_{\leq}^{fin}$.

PROPOSITION 12 *The closure algorithm F can be implemented (for $FT_{\leq}$ and $FT_{\leq}^{fin}$, on-line and off-line) such that it terminates in time $O(n^3)$ where n is the size of the input constraint.*

Proof: We organize the algorithm as a reduction relation on agenda-store-pairs or false. An *agenda* is a finite multiset of atomic constraints and a *store* a constraint φ satisfying the following conditions:

1. For every x and f there exists at most one variable y such that $x[f]y$ belongs to φ.

2. For every x there exists at most one node label a such that $a(x)$ belongs to the store.

3. Every constraint belongs to the store at most once.

The first condition is crucial since it allows us to store selection constraints $x[f]y$ in a table of quadratic size. The idea is that we never have to add two constraints $x[f]y_1$ and $x[f]y_2$ to the store. Instead, we add the first of these constraints plus the consequences $y_1 \leq y_2$ and $y_2 \leq y_1$ which are derivable with F2.

Let φ_0 be the input constraint. Initially, the agenda contains all atomic constraints in φ_0 (which may be fed incrementally in the online case). The initial store contains the constraint $\bigwedge\{x \leq x \mid x \in V(\varphi_0)\} \wedge \bigwedge\{x \sim x \mid x \in V(\varphi_0)\}$. Reduction preserves the invariant that the conjunction of the agenda-store pair is always *complete* in that all one-step consequences of the store with respect to F do either belong to the store itself or the agenda. Also, all agenda-store pairs computed by the algorithm started with φ_0 are equivalent (when considered as a conjunction of constraints).

The algorithm terminates if false is derived or if the agenda becomes empty. In the first case, the input constraint φ_0 is proved unsatisfiable, and in the second one, the final store contains an F-closed constraint equivalent to φ_0. It may happen, however, that the final store differs from the F-closure of φ_0 since the store does not contain atomic constraints multiply; more importantly, it also does not contain all selection constraints belonging to φ_0. However, the atomic constraints of φ_0 which are missing in the final store can be added to it a posteriori without losing F-closedness.

Reduction can be implemented by iteratively executing the following sequence of instructions, which we call *the loop* in the sequel:

1. Select and delete an atomic constraint μ from the agenda. If μ is already in the store then skip.

2. If μ is not already in the store then do the following:

   (A) For every rule scheme F of the form $\varphi \to \mu'$, compute all instances of the form $\mu \wedge \varphi' \to \mu'$ such that φ' belongs to the store. For all these μ' do the following: Test whether μ' is new, meaning that $\mu \neq \mu'$ and μ' does not belong to the agenda nor to the store. If μ' is new then add it to the agenda, otherwise skip.

   (B) If there exists a scheme in F which has an instance of the form $\mu \wedge \varphi \to$ false such that φ is in the store then return false and exit the loop.

3. If μ is an atomic ordering, labeling, or compatibility constraint then add μ to the store. If μ is a selection constraint $x[f]y$ such that the store does not contain $x[f]z$ for all z, then add μ to the store; otherwise skip.

We first discuss the necessary data structures for implementing agenda-store pairs in the off-line case, where the input constraint φ_0 is completely known at start time. Then we argue how to lift the result to the on-line case.

The Off-line Case. Let n be the size of the input constraint φ_0, n_v be the number of its variables and n_f be the number of its features. The agenda can be implemented such that it provides for the following operations in time $O(1)$.

- select and delete an atomic constraint from the agenda.
- add an atomic constraint to the agenda.
- test membership of atomic ordering, labeling, or compatibility constraints in the agenda.

A simple stack or queue is sufficient for ensuring the first two requirements. The third requirement can be satisfied by using three additional arrays for memorizing respectively the atomic ordering, labeling, or compatibility constraints in the agenda.

The store can be implemented by using four arrays: An array of size n_v for labeling constraints $a(x)$ indexed by x (at most one per variable), a table of size $n_v \cdot n_l$ for the selection constraints $x[f]y$ indexed by x and f (at most one per variable x and feature f), and two tables of size n_v^2 for the constraints $x \leq y$ and $x \sim y$ respectively. The store can support the following operations all in time $O(1)$:

- given x test whether there exists a such that the store contains $a(x)$; in case of success return the unique node label a with this property.
- given x and f test whether there exists y such that $x[f]y$ belongs to the store. In case of success, return the unique variable y with this property.
- test the membership of $x \leq y$ or $x \sim y$ in the store.

The initialization phase of the algorithm needs time $O(n^2)$ for allocating the tables for store and agenda and time $O(n)$ for adding the start-up constraints to the agenda and the store. After initialization, every atomic constraint μ is added at most once to the agenda. Since no new variable is created, a complete run of the algorithm can add at most $2 * n_v^2$ constraints of the forms $x \leq y$ and $x \leq y$ to the agenda. This means that the loop is traversed at most $O(n^2)$ many times.

We next verify that each run of the loop needs at most time $O(n)$. Once this its shown, it follows that the overall run time of the algorithm is bounded by $O(n^3)$. For example, let us compute all possible applications of scheme F1.2 for transitivity $x \leq y \wedge y \leq z \rightarrow x \leq z$ to an atomic constraint $u \leq v$. We may either instantiate $x \leq y$ or $y \leq z$ to $u \leq v$. Both cases are symmetric. In the first case, we have to find all z such that $v \leq z$ belongs to the store. This needs time $O(n_v)$. From the latter set of variables, we have to filter out all those z for which $u \leq z$ is new, i.e. neither in the agenda, nor in the store, nor equal to $v \leq u$. Again this can be done in time $O(n_v)$. Last not least, we add all new ordering constraints to the agenda in time $O(n_v)$.

The arguments for the remaining rules schemes are similar except for the occurs check scheme F6 (which is only needed for $FT_{\leq}^{fin}$). In the off-line case, we can perform the occurs check a posteriori, by a simple graph reachability test. Alternatively, we can introduce new reachability formulas of the form $x \rightsquigarrow y$. A reachability formula $x \rightsquigarrow y$ is supported by a

constraint φ if φ contains a conjunction of the form $\bigwedge_{i=1}^{n} x_i[f_i]y_{i+1} \wedge x_{i+1} \leq y_{i+1}$ such that $x_1 = x$ and $y_{n+1} = y$. Support of reachability formulas can be administrated on the fly when using a table of size $O(n_v^2)$.

The On-line Case. In an on-line algorithm, we can feed the input constraint φ_0 piecewise to the agenda. Note that our algorithm is already insensitive to the order in which primitive constraints are picked from the agenda. The additional complication is that the number n_v and n_f of symbols in φ are not known statically. However, by replacing the static tables and arrays by dynamically extensible hash tables we can still guarantee the complexity estimations on the access operations [18]. Note that an on-line implementation of the occurs check cannot be done a posteriori. An incremental occurs check can, however, be implemented based on the reachability formulas $x \rightsquigarrow y$. So, the algorithm has an *incremental* time complexity of $O(n^3)$, both in case of $FT_{\leq}$ and $FT_{\leq}^{fin}$. ■

9. Proofs

We prove the completeness of the satisfiability and entailment tests presented in section 4 and 5. In particular, we show that every F-closed constraint is satisfiable according to Proposition 4 and that all of them can be saturated as postulated in Lemma 1.

9.1. Path Reachability

In both proofs we need the following notion. For all paths p and two variables x, y we define a *path constraint* of the form $x \leq y[p]$ generalizing the path constraint $x \leq y[f]$ and atomic constraints $y \leq x$. A variable assignment α into $FT_{\leq}$ (resp. $FT_{\leq}^{fin}$) is a solution of $x \leq y[p]$ in the respective structure if $p \in D_{\alpha(x)}$ and $\alpha(y) \leq \alpha(x)[p]$. We generalize the judgments for syntactic support defined so far to $\varphi \vdash y \leq x[p]$, which we read as "$y$ is reachable from x over path p in φ":

$$\begin{array}{lll} \varphi \vdash y \leq x[\varepsilon] & \text{if} & y \leq x \text{ in } \varphi \\ \varphi \vdash y \leq x[f] & \text{if} & x[f]y \text{ in } \varphi, \\ \varphi \vdash y \leq x[pq] & \text{if} & \varphi \vdash z \leq x[p] \text{ and } \varphi \vdash y \leq z[q] \text{ for some } z. \end{array}$$

We also need path constraints of the form $a(x[p])$ a solution of which is a variable assignment α satisfying $(p, a) \in L_{\alpha(x)}$. Again, we need a notion of syntactic support $\varphi \vdash a(x[p])$ which reads as "the node label a is reachable in φ from x over path p":

$$\varphi \vdash a(x[p]) \quad \text{if} \quad \varphi \vdash y \leq x[p] \text{ and } a(y) \text{ in } \varphi \text{ for some } y,$$

For example, if φ is the constraint $x \leq y \wedge a(y) \wedge x[f]u \wedge x[g]z \wedge z[f]x \wedge b(z)$ then the following reachability propositions hold: $\varphi \vdash x \leq y[\varepsilon]$, $\varphi \vdash z \leq x[g]$, $\varphi \vdash x \leq y[gf]$, $\varphi \vdash x \leq x[gf]$, *etc.*, as well as $\varphi \vdash a(y[\varepsilon])$, $\varphi \vdash b(x[g])$, $\varphi \vdash b(x[gfg])$, *etc.*

LEMMA 4 *If* $\varphi \vdash z{\leq}x[fp]$ *holds then there exist variables* x', y' *such that* $\varphi \vdash x'{\leq}x[\varepsilon]$, $x'[f]y' \in \varphi$, *and* $\varphi \vdash z{\leq}y'[p]$.

9.2. Completeness of the Satisfiability Test

We next prove the following proposition stated without proof in Section 4.

PROPOSITION 4 *Every* $\mathsf{F1}-\mathsf{F5}$ *closed constraint is satisfiable over* $FT_{\leq}$*; every* $\mathsf{F1}-\mathsf{F6}$ *closed constraint is satisfiable over* $FT_{\leq}^{fin}$.

Proof: The proof is in four steps elaborated in this section. First, we define a syntactic property of a constraint, called path consistency. Second, we argue that a path consistent constraint is satisfiable if is $\mathsf{F1}-\mathsf{F2}$-closed (Lemma 5). Third, we show that an $\mathsf{F3}$-$\mathsf{F5}$-closed constraint is path consistent (Lemma 6). In the last step, we verify that the solution constructed in step two is finite for $\mathsf{F6}$-closed constraints. ■

DEFINITION (PATH CONSISTENCY) *We call a constraint* φ path consistent *if the following two conditions hold for all x, y, p, a, and b.*

1. *If* $\varphi \vdash a(x[p])$ *and* $\varphi \vdash b(x[p])$ *then* $a = b$.
2. *If* $\varphi \vdash a(x[p])$, $x{\sim}y$ in φ, *and* $\varphi \vdash b(y[p])$ *then* $a = b$.

Apparently, condition 2 implies condition 1 for $\mathsf{F3.1}$-closed constraints. We require the first condition nevertheless, since we which to split the proof into two lemmas (Lemma 5 and Lemma 6) where we assume only $\mathsf{F1}$-$\mathsf{F2}$-closedness for the first lemma.

LEMMA 5 *Every* $\mathsf{F1}$-$\mathsf{F2}$*-closed and path consistent constraint* φ *is satisfiable in* $FT_{\leq}$*; if* φ *is* $\mathsf{F6}$*-closed in addition then it is also satisfiable in* $FT_{\leq}^{fin}$.
Furthermore, the following variable assignment $\mathsf{min}_{\varphi}(.)$ *is the least solution of an* F*-closed constraint* φ. *For all* $x \in \mathcal{V}(\varphi)$*:*

$$\begin{aligned} D_{\mathsf{min}_{\varphi}(x)} &= \{p \mid \varphi \vdash y{\leq}x[p] \text{ for some } y\} \\ L_{\mathsf{min}_{\varphi}(x)} &= \{(p,a) \mid \varphi \vdash a(x[p])\} \end{aligned}$$

Proof: Let φ be $\mathsf{F1}$-$\mathsf{F2}$-closed and path consistent. The first condition of path consistency implies that $L_{\mathsf{min}_{\varphi}(x)}$ is a partial function. Thus $\mathsf{min}_{\varphi}(x)$ is a feature tree in $FT_{\leq}$.
If φ is also $\mathsf{F6}$-closed then it is not possible that $\varphi \vdash x{\leq}x[p]$ holds for some path $p \neq \varepsilon$. Hence, for all x, y, p with $\varphi \vdash y{\leq}x[p]$ it holds that the length of p is bounded by the number of variables in φ (since for each prefix q of p there must be a distinct variable z such that $\varphi \vdash z{\leq}x[q]$). Thus, if φ is $\mathsf{F6}$-closed then $\mathsf{min}_{\varphi}(x)$ belongs to $FT_{\leq}^{fin}$.
We next verify that min_{φ} is a solution of φ in $FT_{\leq}$ i.e. that min_{φ} is a solution of all atomic constraints in φ:

- Let $y{\leq}x$ *in* φ. For all z, if $\varphi \vdash z{\leq}y[p]$ then $\varphi \vdash z{\leq}x[p]$ by the definition of syntactic support. Thus, $D_{\mathsf{min}_\varphi(y)} \subseteq D_{\mathsf{min}_\varphi(x)}$. For all a if $\varphi \vdash a(y[p])$ then $\varphi \vdash a(x[p])$ by the definition of syntactic support. Thus, $L_{\mathsf{min}_\varphi(y)} \subseteq L_{\mathsf{min}_\varphi(x)}$, *i.e.*, $\mathsf{min}_\varphi(y){\leq}\mathsf{min}_\varphi(x)$.

- Consider $x[f]y$ *in* φ. We prove the following equivalences for all p, z, and b:

$$\varphi \vdash z{\leq}x[fp] \;\text{ iff }\; \varphi \vdash z{\leq}y[p] \qquad \text{and} \qquad \varphi \vdash b(x[fp]) \;\text{ iff }\; \varphi \vdash b(y[p])$$

  The first equivalence implies $D_{\mathsf{min}_\varphi(y)} = \{p \mid fp \in D_{\mathsf{min}_\varphi(x)}\}$ and the second one is equivalent to $L_{\mathsf{min}_\varphi(y)} = \{(p,b) \mid (fp,b) \in L_{\mathsf{min}_\varphi(x)}\}$. We start by proving the first equivalence. If $\varphi \vdash z{\leq}y[p]$ then $\varphi \vdash z{\leq}x[fp]$ since $x[f]y$ *in* φ. Suppose $\varphi \vdash z{\leq}x[fp]$. By Lemma 4 there exists x' and y' such that

$$\varphi \vdash x'{\leq}x[\varepsilon], \qquad x'[f]y' \text{ in } \varphi, \qquad \varphi \vdash z{\leq}y'[p].$$

  The F1.2-closedness of φ and $\varphi \vdash x'{\leq}x[\varepsilon]$ implies $x'{\leq}x$ *in* φ. The F2-closedness ensures $y'{\leq}y$ *in* φ such that $\varphi \vdash z{\leq}y[p]$ holds. We now prove the second equivalence above. If $\varphi \vdash b(x[fp])$ then there exists z such that $\varphi \vdash z{\leq}x[fp]$ and $b(z) \in \varphi$. The first equivalence implies $\varphi \vdash z{\leq}y[p]$ and thus $\varphi \vdash b(y[p])$. The converse is analogous.

- Let $a(x)$ in φ. Reflexivity (F1.1-closedness) implies $x{\leq}x$ *in* φ. Thus $\varphi \vdash a(x[\varepsilon])$ and hence $(\varepsilon, a) \in L_{\mathsf{min}_\varphi(x)}$.

- Let $x{\sim}y$ in φ. We have to show that the set $L_{\mathsf{min}_\varphi(x)} \cup L_{\mathsf{min}_\varphi(y)}$ is a partial function. If $(p,a) \in L_{\mathsf{min}_\varphi(x)}$ and $(p,b) \in L_{\mathsf{min}_\varphi(y)}$ then $\varphi \vdash a(x[p])$ and $\varphi \vdash b(y[p])$. The second condition of path consistency for φ implies $a = b$. ■

LEMMA 6 *Every* F3-F5*-closed constraint is path consistent.*

Proof: Let φ be F3, F4, F5-closed. As mentioned before, the first condition of path consistency follows from the second one and F3.1-closedness. The proof of the second condition is by induction on paths p. We assume x, y, a, and b such that $\varphi \vdash a(x[p])$, $x{\sim}y$ *in* φ, and $\varphi \vdash b(y[p])$. If $p = \varepsilon$, then there exist $n, m \geq 0$, $x_1, \ldots, x_n$, $y_1, \ldots y_m$ such that:

$$\begin{aligned} & a(x_n) \wedge x_n{\leq}x_{n-1} \wedge \ldots \wedge x_1{\leq}x \text{ in } \varphi\,, \\ \text{and } \; & b(y_m) \wedge y_m{\leq}y_{m-1} \wedge \ldots \wedge y_1{\leq}y \text{ in } \varphi\,. \end{aligned}$$

F3-closedness implies that $x_n{\sim}y_m$ *in* φ (F3.2 yields $x{\sim}y_1$ *in* φ, ..., $x{\sim}y_m$ *in* φ. Therefore $y_m{\sim}x$ *in* φ by F3.3-closedness, and hence $y_m{\sim}x_1$ *in* φ, ..., $y_m{\sim}x_n$ *in* φ by F3.2-closedness.) Hence, F5-closedness implies $a = b$.

In the case $p = gp'$, Lemma 4 yields the existence of x', y', $\tilde{x}$, and $\tilde{y}$ such that:

$$\begin{aligned} & \varphi \vdash x'{\leq}x[\varepsilon], \quad x'[g]\tilde{x} \text{ in } \varphi\,, \quad \varphi \vdash a(\tilde{x}[p'])\,, \\ \text{and } \; & \varphi \vdash y'{\leq}y[\varepsilon], \quad y'[g]\tilde{y} \text{ in } \varphi\,, \quad \varphi \vdash b(\tilde{y}[p'])\,. \end{aligned}$$

Since $x{\sim}y$ *in* φ we have $x'{\sim}y'$ *in* φ by F3-closedness (as above). Thus, F4-closedness implies $\tilde{x}{\sim}\tilde{y}$ *in* φ such that $a{=}b$ follows by induction hypothesis (from $\varphi \vdash a(\tilde{x}[p'])$, $\varphi \vdash b(\tilde{y}[p'])$ and $\tilde{x}{\sim}\tilde{y}$ *in* φ). ■

Lemmas 5 and 6 yield a further result on entailment which may be of its own interest.

COROLLARY 2 *Let φ be an* F*-closed constraint. Then $\varphi \models \exists y(x[f]y)$ if and only if there exists a variable z such that $\varphi \vdash z{\leq}x[f]$.*

Proof: Assume that $\varphi \vdash z{\leq}x[f]$ does not hold for all z. According to Lemmas 6 and 5n it holds for the least solution min_φ of an F-closed constraint that $f \notin D_{\mathsf{min}_\varphi(x)}$. Hence $\varphi \not\models \exists y(x[f]y)$. ■

9.3. Saturation

We prove the existence of a saturated formula $\mathrm{Sat}(\varphi)$ as postulated in Lemma 1. This formula contradicts all (relevant) atomic constraints not entailed by φ simultaneously.

We construct $\mathrm{Sat}(\varphi)$ by means of two operators Γ_1 and Γ_2 on constraints. The operator Γ_2 is such that $\Gamma_2(\varphi)$ disentails all atomic constraints μ of the forms $x{\sim}y$, $x{\leq}y$, and $a(x)$ (but not selection constraints) which are not syntactically supported in φ (Lemma 9). The operator Γ_1 is necessary to also disentail selection constraints. Given a constraint φ, $\Gamma_1(\varphi)$ extends φ such that $\Gamma_2(\Gamma_1(\varphi))$ disentails all relevant μ. In a sense, Γ_1 is a "preprocessor" for Γ_2.

DEFINITION OF Γ_1 *Let φ be a constraint. For all $x \in V(\varphi)$ and $f \in F(\varphi)$ let v_{xf} be a fresh variable. Depending on this choice of variables, we define $\Gamma_1(\varphi)$ to be the following* F*-closure whenever it exists and* false *otherwise.*

$$\Gamma_1(\varphi) \quad = \quad cl(\varphi \wedge \bigwedge\{x[f]v_{xf} \mid x \in V(\varphi) \text{ and } f \in F(\varphi)\})$$

DEFINITION OF Γ_2 *Let φ be a constraint. Let v_1 and v_2 be distinct fresh variables, a_1 and a_2 be distinct labels, and for every pair of variables $x,y \in V(\varphi)$ and $f \in L(\varphi)$ let v_x be a fresh variable and let f_x and f_{xy} be fresh features. We define a first-order formula $\Gamma_2(\varphi)$ depending on $v_1, v_2, a_1, a_2, f_x, f_{xy}$, and v_x as follows:*

$$\begin{aligned}
\Gamma_2(\varphi) \quad = \quad & \varphi \ \wedge \ \bigwedge\{x[f_x]v_x \wedge \neg\exists y'(y[f_x]y') \mid \varphi \not\vdash x{\leq}y,\ x,y \in V(\varphi)\} && (1)\\
& \wedge \ \bigwedge\{x[f_{xy}]v_1 \wedge y[f_{xy}]v_2 \mid \varphi \not\vdash x{\sim}y,\ x,y \in V(\varphi)\} && (2)\\
& \wedge \ \bigwedge\{x{\sim}v_1 \wedge x{\sim}v_2 \mid \text{for all } a \in L:\ \varphi \not\vdash a(x),\ x \in V(\varphi)\} && (3)\\
& \wedge \ a_1(v_1) \wedge a_2(v_2) && (4)
\end{aligned}$$

EXAMPLE 8 *For illustration of Γ_1 and Γ_2 consider the constraint φ equal to $x[f]x \wedge y{\leq}x$ which does not entail $x[f]y$ if we assume $x \neq y$. The constraint φ can be* F*-closed for $FT_{\leq}$ (but not for $FT_{\leq}^{fin}$) by adding the following trivial atomic constraints: $x{\sim}y \wedge y{\sim}x \wedge x{\sim}x \wedge y{\sim}y \wedge x{\leq}x \wedge y{\leq}y$ which we omit for sake of simplicity. In order to disentail $x[f]y$ we first*

compute $\Gamma_1(\varphi)$ *by adding* $x[f]v_{xf}$ *and* $y[f]v_{yf}$ *to* φ *and computing the* F*-closure. Now,* $\Gamma_1(\varphi)$ *is (up to trivial and compatibility constraints):*

$$\Gamma_1(\varphi) \;=\; \begin{cases} x[f]x \wedge y \leq x \wedge x[f]v_{xf} \wedge y[f]v_{yf} \wedge \\ v_{yf} \leq v_{xf} \wedge v_{xf} \leq x \wedge x \leq v_{xf} \wedge y \leq v_{xf} \wedge v_{yf} \leq x \end{cases}$$

Observe that $\Gamma_1(\varphi)$ *does not contain* $v_{xf} \leq y$*; that is,* $\Gamma_1(\varphi) \not\vdash v_{xf} \leq y$*. Now* $\Gamma_2(\Gamma_1(\varphi))$ *disentails* $v_{xf} \leq y$ *due to clause* (1) *which asserts that* v_{xf} *allows selection at feature* $f_{v_{xf}}$ *while* y *does not (since* $v_{xf}[f_{v_{xf}}]v_{v_{xf}} \wedge \neg\exists z(v_{yf}[f_{v_{yf}}]z)$ *in* $\Gamma_2(\Gamma_1(\varphi))$*). Hence,* $\Gamma_2(\Gamma_1(\varphi))$ *also disentails* $x[f]y$*.*

Note that Example 8 does also illustrates why a two step saturation procedure is needed: The key idea is that the feature $f_{v_{xf}}$ allows to contradict entailment of $x[f]y$ for all $y \in V(\varphi)$ such that $\varphi \not\vdash x[f]y$. This feature $f_{v_{xf}}$ is introduced in the second step on the basis of the variable v_{xf} which is added freshly in the first step.

LEMMA 7 (PROPERTIES OF Γ_1) *Let* φ *be an* F*-closed (and hence satisfiable) constraint. Then* $\Gamma_1(\varphi)$ *is satisfiable and satisfies the following two properties for all atomic constraints* μ*, variables* x,y *and features* f*:*

(P1) *If* $\varphi \not\vdash \mu$*, and* $V(\mu) \subseteq V(\varphi)$*, then* $\Gamma_1(\varphi) \not\vdash \mu$*.*

(P2) *If* $x,y \in V(\varphi)$*,* $f \in F(\varphi)$*, and* $\varphi \not\vdash x[f]y$*, then* $\Gamma_1(\varphi) \not\vdash y \leq v_{xf}$ *or* $\Gamma_1(\varphi) \not\vdash v_{xf} \leq y$*.*

Proof: We first show that $\Gamma_1(\varphi)$ is satisfiable and then show (P1) and (P2). For proving the satisfiability of $\Gamma_1(\varphi)$, we give an inductive construction of $\Gamma_1(\varphi)$ and show that all constraints in this construction are satisfiable. Let n be the cardinality of the set $V = \{v_{xf} \mid x \in V(\varphi),\ f \in F(\varphi)\}$ and fix an enumeration $var : \{1,\ldots,n\} \to V$, i.e. var is a function that is one-to-one and onto. Then, we consider the following sequence of constraints for $1 \leq i \leq n$:

$$\begin{array}{lcll} \varphi_0 & = & \varphi & \\ \varphi_i & = & \mathrm{cl}(\varphi_{i-1} \wedge x[f]v_{xf}) & \text{if } var(i) = v_{xf} \end{array}$$

Of course, we have to show that all of the above closures exist. If so then, apparently, it follows that $\Gamma_1(\varphi) = \varphi_n$. We show the existence of the closures in the definition of φ_i by induction on i. For the induction step, we assume for $0 < i \leq n$ with $var(i) = v_{xf}$ that φ_{i-1} exists, and show that the constraint $\widetilde{\varphi}_i$ defined below is F-closed. Since $\widetilde{\varphi}_i$ contains $\varphi_i \wedge x[f]v_{xf}$ this shows that the closure $\mathrm{cl}(\varphi_{i-1} \wedge x[f]v_{xf})$ exists.

$$\widetilde{\varphi}_i \;=\; \varphi_{i-1} \wedge\ x[f]v_{xf} \wedge v_{xf} \leq v_{xf} \wedge v_{xf} \sim v_{xf} \tag{4.1}$$

$$\wedge\ \bigwedge\{z \leq v_{xf} \mid \varphi_{i-1} \vdash z \leq x[f]\} \tag{4.2}$$

$$\wedge\ \bigwedge\{v_{xf} \leq z \mid \varphi_{i-1} \vdash x?[f] \leq z\} \tag{4.3}$$

$$\wedge\ \bigwedge\{v_{xf} \sim z \wedge z \sim v_{xf} \mid \text{ex. } y : \varphi_{i-1} \vdash x?[f] \leq y \text{ and } y \sim z \text{ in } \varphi_{i-1}\} \tag{4.4}$$

$$\wedge\ \bigwedge\{v_{xf} \sim z \wedge z \sim v_{xf} \mid \text{ex. } y : \varphi_{i-1} \vdash z \leq y[f] \text{ and } y \sim x \text{ in } \varphi_{i-1}\} \tag{4.5}$$

It is clear that $\widetilde{\varphi}_i$ is contained in φ_i, hence it suffices to show that v_{xf} is F-closed. The F-closedness of $\widetilde{\varphi}_i$ is proved by a case distinction over the rules schemes in F. We have only to consider those instances of schemes in F which contains the new variable v_{xf}. For sake of readability, we allow us to also denote variables with u, v, w.

F1.1 Reflexivity of the ordering relation holds since $v_{xf} \leq v_{xf}$ *in* $\widetilde{\varphi}_i$ by clause (4.1).

F1.2 We assume $u \leq v$ *in* $\widetilde{\varphi}_i$ and $v \leq w$ *in* $\widetilde{\varphi}_i$ and show that $u \leq w$ *in* $\widetilde{\varphi}_i$. We make a case distinction depending on which of the variables u, v, w equal v_{xf}.

If $u, v, w \neq v_{xf}$**,** then F1.2-closedness of φ_{i-1} yields $u \leq w$ *in* φ_{i-1}. Thus $u \leq w$ *in* $\widetilde{\varphi}_i$.

If $u = w = v_{xf}$**,** then $u \leq w$ *in* $\widetilde{\varphi}_i$ iff $v_{xf} \leq v_{xf}$ *in* $\widetilde{\varphi}_i$, and this follows from clause (4.1).

If $u = v = v_{xf}$**,** then $u \leq w$ *in* $\widetilde{\varphi}_i$ iff $v_{xf} \leq w$ *in* $\widetilde{\varphi}_i$, and this follows from $v \leq w$ *in* $\widetilde{\varphi}_i$.

If $v = w = v_{xf}$**,** then $u \leq w$ *in* $\widetilde{\varphi}_i$ iff $u \leq v_{xf}$ *in* $\widetilde{\varphi}_i$, and this follows form $u \leq v$ *in* $\widetilde{\varphi}_i$.

If $u = v_{xf}$ **and** $v, w \neq v_{xf}$**,** then $v_{xf} \leq v$ *in* $\widetilde{\varphi}_i$ and hence $\varphi_{i-1} \vdash x?[f] \leq v$ by clause (4.3). By F1.2-closedness of φ_{i-1} (transitivity) it follows that $\varphi_{i-1} \vdash x?[f] \leq w$ and hence, by clause (4.3) again, $v_{xf} \leq w$ *in* $\widetilde{\varphi}_i$, i.e. $u \leq w$ *in* $\widetilde{\varphi}_i$.

If $w = v_{xf}$ **and** $u, v \neq v_{xf}$**.** This case is symmetric to the previous one when using clause (4.2) instead of clause (4.3).

If $v = v_{xf}$ **and** $u, w \neq v_{xf}$**,** then, by clauses (4.2) and (4.3), $\varphi_{i-1} \vdash u \leq x[f]$ and $\varphi_{i-1} \vdash x?[f] \leq w$. By F-closedness of φ_{i-1} (transitivity and descent, F1.2 and F2) it follows that $u \leq w$ *in* φ_{i-1} and hence $u \leq w$ *in* $\widetilde{\varphi}_i$.

F2 We assume $u[g]u'$ *in* $\widetilde{\varphi}_i$, $u \leq v$ *in* $\widetilde{\varphi}_i$ and $v[g]v'$ *in* $\widetilde{\varphi}_i$ and show $u' \leq v'$ *in* $\widetilde{\varphi}_i$. If $u, v, u', v' \in V(\varphi_{i-1})$ then this follows from the F2-closedness of φ_{i-1}. Otherwise, at least one of these variables u, v, u', v' is equal to the new variable v_{xf}. Since $x[f]v_{xf}$ is the only selection constraint added to φ_{i-1}, it follows that $v_{xf} \notin \{u, v\}$. Hence, $v_{xf} \in \{u', v'\}$.

If $v_{xf} = u'$**,** then $x = u$ and $g = f$.

If $v_{xf} = v'$**,** then $u' \leq v'$ *in* $\widetilde{\varphi}_i$ follows from the F1.1-closedness of $\widetilde{\varphi}_i$ (reflexivity).

If $v_{xf} \neq v'$**,** then $\varphi_{i-1} \vdash x?[f] \leq v'$ and hence it follows from clause (4.3) that $v_{xf} \leq v'$ *in* $\widetilde{\varphi}_i$, i.e. $u' \leq v'$ *in* $\widetilde{\varphi}_i$.

If $v_{xf} = v'$**,** then $x = v$ and $g = f$.

If $v_{xf} = u'$**,** then $u' \leq v'$ *in* $\widetilde{\varphi}_i$ follows from the F1.1-closedness of $\widetilde{\varphi}_i$ (reflexivity).

If $v_{xf} \neq u'$**,** then $\varphi_{i-1} \vdash u' \leq x[f]$ and hence it follows from clause (4.2) that $u' \leq v_{xf}$ *in* $\widetilde{\varphi}_i$, i.e. $u' \leq v'$ *in* $\widetilde{\varphi}_i$.

F3.1 Reflexivity of the compatibility relation holds since $v_{xf} \sim v_{xf}$ *in* $\widetilde{\varphi}_i$ by clause (4.1).

F3.2 Assume $u \leq v$ *in* $\widetilde{\varphi}_i$ and $v \sim w$ *in* $\widetilde{\varphi}_i$. We have to show that $u \sim w$ *in* $\widetilde{\varphi}_i$.

If $u, v, w \neq v_{xf}$**,** then $u \sim w$ *in* $\widetilde{\varphi}_i$ follows from F-closedness of φ_{i-1}.

If $u = v = v_{xf}$**,** then $v \sim w$ *in* $\widetilde{\varphi}_i$ iff $v_{xf} \sim w$ *in* $\widetilde{\varphi}_i$ iff $u \sim w$ *in* $\widetilde{\varphi}_i$.

If $u = w = v_{xf}$**,** then $u \sim w$ *in* $\widetilde{\varphi}_i$ follows from clause (4.1).

If $v = w = v_{xf}$ **and** $u \neq v_{xf}$**,** then, by clause (4.2) $\varphi_{i-1} \vdash u{\leq}x[f]$. By F-closedness of φ_{i-1} we have $x{\sim}x$ *in* φ_{i-1} and hence, by clause (4.5), $u{\sim}v_{xf}$ *in* $\widetilde{\varphi}_i$. Hence $u{\sim}v$ *in* $\widetilde{\varphi}_i$.

If $u = v_{xf}$ **and** $v, w \neq v_{xf}$**,** then by clause (4.3), $\varphi_{i-1} \vdash x?[f]{\leq}v$ and hence, by clause (4.4), $v_{xf}{\sim}w$ *in* $\widetilde{\varphi}_i$, i.e $u{\sim}w$ *in* $\widetilde{\varphi}_i$.

If $w = v_{xf}$ **and** $u, v \neq v_{xf}$**,** then $v{\sim}w$ is equal to $v{\sim}v_{xf}$ and could have been added by clause (4.4) or clause (4.5).

(4.4) Then, by clause (4.4), exists v' such that $\varphi_{i-1} \vdash x?[f]{\leq}v'$ and $v'{\sim}v$ *in* φ_{i-1}. By F3.2 and F3.3-closedness of φ_{i-1} it holds that $v'{\sim}u$ *in* φ_{i-1} and hence $u{\sim}v_{xf}$ *in* $\widetilde{\varphi}_i$ by clause (4.4) again, i.e. $u{\sim}w$ *in* $\widetilde{\varphi}_i$.

(4.5) Then, by clause (4.5), exists x' such that $\varphi_{i-1} \vdash v{\leq}x'[f]$ and $x'{\sim}x$ *in* φ_{i-1}. By F1.2-closedness of φ_{i-1} (transitivity), $\varphi_{i-1} \vdash u{\leq}x'[f]$ so that $u{\sim}v_{xf}$ *in* $\widetilde{\varphi}_i$ by clause (4.5) again, i.e. $u{\sim}w$ *in* $\widetilde{\varphi}_i$.

If $v = v_{xf}$ **and** $u, w \neq v_{xf}$**,** then $v_{xf}{\sim}w$ could have been added by clause (4.4) or clause (4.5). The argument is similar to the previous one.

F3.3 Symmetry of the compatibility relation holds since when ever a compatibility constraint is added in either (4.1), (4.2), or (4.3) then also its symmetric variant is added.

F4 We assume $u[g]u'$ *in* $\widetilde{\varphi}_i$, $u{\sim}v$ *in* $\widetilde{\varphi}_i$, and $v[g]v'$ *in* $\widetilde{\varphi}_i$ and show $u'{\leq}v'$ *in* $\widetilde{\varphi}_i$. Because of the F4-closedness of φ_{i-1} this holds trivially if $u, v, u', v' \in V(\varphi_{i-1})$. Otherwise, there exists at least one of these variables which is equal to the new variable v_{xf}. Since $x[f]v_{xf}$ is the unique selection constraint added to φ_{i-1} it follows that $v_{xf} \notin \{u, v\}$. Hence, $v_{xf} \in \{u', v'\}$. We can assume without loss of generality that $v_{xf} = u'$ since due to symmetry (F3.3-closedness of φ_{i-1}) it holds that $v{\sim}u$ *in* φ_{i-1} such that the rles of u and v can be exchanged. So we assume $v_{xf} = u'$, $x = u$, and $g = f$.

If $v_{xf} = v'$**,** then $u'{\sim}v' \in \widetilde{\varphi}_i$ follows from the F3.1-closedness of $\widetilde{\varphi}_i$ (reflexivity).

If $v_{xf} \neq v'$**,** then $\varphi_{i-1} \vdash v'{\leq}v[f]$ and $x{\sim}v$ *in* φ_{i-1}. Since φ_{i-1} is F3.3-closed we also have $v{\sim}x$ *in* φ_{i-1}. Hence it follows from clause (4.5) that $v_{xf}{\leq}v'$ *in* $\widetilde{\varphi}_i$, i.e. $u'{\leq}v'$ *in* $\widetilde{\varphi}_i$.

F5 The clash axiom F5 does not apply to $\widetilde{\varphi}_i$ for two reasons: No compatibility constraint $y{\sim}z$ has been added to φ_{i-1} for some variables $y, z \in V(\varphi_{i-1})$, and no labeling constraint has been added for the new variable v_{xf}.

F6 For the case of $FT^{fin}_{\leq}$, we show that if $\widetilde{\varphi}_i$ is not F6-closed then φ_{i-1} is also not F6-closed. Suppose that $\widetilde{\varphi}_i$ is not F6-closed. Hence, there exists a cyclic constraint of the form $\bigwedge_{j=1}^{n} x_j[f_j]y_{j+1} \wedge x_{j+1}{\leq}y_{j+1}$ in $\widetilde{\varphi}_i$ where $x_{n+1} = x_1$ and $n \geq 1$. If $x_j, y_j \in V(\varphi_{i-1})$ for all $1 \leq j \leq n+1$ then, of course, φ_{i-1} is not F6-closed and we are done. Otherwise, there exists $1 \leq j \leq n+1$ such that $y_j = v_{xf}$ (it is not possible that $x_j = v_{xf}$ since not $v_{xf}[g]z$ *in* $\widetilde{\varphi}_i$ for all z). We can assume without loss of generality that all y_j are distinct (otherwise there exists a shorter cycle in $\widetilde{\varphi}_i$ which can be considered instead). Hence, the index j with $y_j = v_{xf}$ is unique. Without loss of generality, we can assume $j = n+1$ (since we can shift the indexes of the variables in the cycle). From the definition of

$\widetilde{\varphi}_i$ and the fact that $x_{n+1} \leq v_{xf}$ *in* φ_i it follows that $\varphi_{i-1} \vdash x_{n+1} \leq x_n[f_n]$. The definition of syntactic support together with $\mathsf{F1.1} - \mathsf{F1.2}$-closedness of φ_{i-1} yields the existence of x'_n and y'_{n+1} such that $x_{n+1} \leq y'_{n+1} \wedge x'_n[f_n]y'_{n+1} \wedge x'_n \leq x_n$ *in* φ_{i-1}. This implies the existence of the following cycle in φ_{i-1} which shows that φ_{i-1} is not $\mathsf{F6}$-closed:

$$(\bigwedge_{j=1}^{n-2} x_j[f_j]y_{j+1} \wedge x_{j+1} \leq y_{j+1}) \wedge (x_{n-1}[f_{n-1}]y_n \wedge x'_n \leq y_n) \wedge (x'_n[f_n]y'_{n+1} \wedge x_{n+1} \leq y'_{n+1})$$

Now we check properties (P1) and (P2) claimed in Lemma 7, both by contraposition.

(P1) Assume that $\Gamma_1(\varphi) \vdash \mu$, and $V(\mu) \subseteq V(\varphi)$. We show that $\varphi \vdash \mu$ by case distinction over the forms of atomic constraints μ.

- $\mu = x \leq y$ or $\mu = x \sim y$**:** If $\Gamma_1(\varphi) \vdash \mu$ then μ *in* $\Gamma_1(\varphi)$ or $x = y$. If $x = y$, then trivially $\varphi \vdash \mu$. Otherwise, if μ *in* $\Gamma_1(\varphi)$. From $V(\mu) \subseteq V(\varphi)$ and the concrete representation of $\Gamma_1(\varphi)$ coming with $\Gamma_1(\varphi) = \varphi_n$, we can deduce μ *in* φ. Hence $\varphi \vdash \mu$.
- $\mu = a(x)$**:** If $\Gamma_1(\varphi) \vdash a(x)$ then there exists a variable x' such that $a(x') \wedge x' \leq x$ *in* $\Gamma_1(\varphi)$. Since labeling constraints are not added by the closure operation one obtains that $a(x')$ *in* φ. The assumption $V(\mu) \subseteq V(\varphi)$ gives $x \in V(\varphi)$ and hence $V(x' \leq x) \subseteq V(\varphi)$. As already proved in the previous case, this implies $\varphi \vdash x' \leq x$. Hence, we conclude $\varphi \vdash a(x)$.
- $\mu = x[f]y$**:** If $\Gamma_1(\varphi) \vdash x[f]y$ then there exist variables u, u' and v, v' such that:

  $$\Gamma_1(\varphi) \vdash \mu' \text{ for all } \mu' \in \{u \leq x, x \leq v, y \leq u', v' \leq y\},$$
  $$\text{and } u[f]u' \wedge v[f]v' \text{ in } \Gamma_1(\varphi).$$

  By assumption, $x, y \in V(\mu) \subseteq V(\varphi)$. Also $u, v \in V(\varphi)$ holds since $\Gamma_1(\varphi) = \varphi_n$ contains no selection constraint of the form $z_1[f]v$ where $z_1 \notin V(\varphi)$.

  In the case $u', v' \in V(\varphi)$, it follows easily that $\varphi \vdash x[f]y$. We can without loss of generality assume that $u', v' \in V(\varphi)$. To see why, suppose $u' \notin V(\varphi)$. Then $u' = v_{uf}$ by construction of $\Gamma_1(\varphi) = \varphi_n$: Let $var(v_{uf}) = i$. Then by Clause (4.2) $\varphi_{i-1} \vdash y \leq u[f]$ which means that there must exist variables $w, w' \in V(\varphi_{i-1})$ such that $y \leq w' \wedge w[f]w' \wedge w \leq x$ *in* φ_{i-1}. Hence, we can replace w, w' for u, u' above and obtain the same situation up to renaming. By induction over $var(v_{uf})$ we find a replacement for u' in $V(\varphi)$. The argument for v' is dual.

(P2) Assume that $\Gamma_1(\varphi) \vdash z \leq v_{xf}$ and $\Gamma_1(\varphi) \vdash v_{xf} \leq z$ for some variable $x \in V(\varphi)$ and $f \in F(\varphi)$. Then by clauses (4.2) and (4.3) there must exist variables $y, y', u, u' \in V(\Gamma_1(\varphi))$ such that $\Gamma_1(\varphi) \vdash z \leq x[f]$ and $\Gamma_1(\varphi) \vdash x?[f] \leq z$. By definition of syntactic support these assumptions imply $\Gamma_1(\varphi) \vdash x[f]z$ and hence, by case (1) above, $\varphi \vdash x[f]z$. ■

LEMMA 8 (Γ_2 PRESERVES SATISFIABILITY) *If* φ *is* F*-closed, then* $\Gamma_2(\varphi)$ *is satisfiable.*

Proof: Let φ_Γ be the constraint part of $\Gamma_2(\varphi)$ (*i.e.*, the conjunction of all atomic constraints in $\Gamma_2(\varphi)$ but without the negative formulas added by clause (1). It is not difficult to show

that φ_Γ is F-closed up to trivial constraints ($x{\leq}x$ and $x{\sim}x$) and symmetric compatibility constraints. Note in particular, that each fresh feature f_x occurs only once in $\Gamma_2(\varphi)$ (and hence neither F2 nor F4 apply), and that the fresh features f_{xy} occur exactly twice in $\Gamma_2(\varphi)$, namely in selections at x and y, for which neither $x{\sim}y$ nor, by F3.1-closedness of φ, $x{\leq}y$ or $y{\leq}x$ occur in φ.

Hence $\mathsf{min}_{\varphi_\Gamma}$ as defined in Lemma 5 of Section 9.2 is a solution of φ_Γ. It suffices to check that $\mathsf{min}_{\varphi_\Gamma}$ also satisfies the negated selection constraints added in clause (1) of $\Gamma_2(\varphi)$.

Assume $\neg\exists y'(y[f_x]y')$ *in* $\Gamma_2(\varphi)$, hence also $x[f_x]v_x$ *in* $\Gamma_2(\varphi)$ and $\varphi \not\vdash x{\leq}y$. F-closedness of φ and $\varphi \not\vdash x{\leq}y$ imply that $\varphi \vdash x{\leq}y[\varepsilon]$. Since f_x has a unique occurrence in $\Gamma_2(\varphi)$, this implies that $\varphi_\Gamma \vdash v_x{\leq}y[f_x]$, and hence $f_x \notin D_{\mathsf{min}_{\varphi_\Gamma}(y)}$. ■

LEMMA 9 (Γ_2 CONTRADICTS NON-SELECTION CONSTRAINTS) *Let φ be an* F*-closed constraint and let μ be an atomic constraint of the form $x{\sim}y$, $x{\leq}y$, or $a(x)$ with $x,y \in V(\varphi)$. Then $\Gamma_2(\varphi) \models \neg\mu$ if and only if $\varphi \not\vdash \mu$.*

Proof: If $\Gamma_2(\varphi) \models \neg\mu$ then $\varphi \not\vdash \mu$ by Lemma 8 and correctness of syntactic support. For the inverse direction we inspect the definition of $\Gamma_2(\varphi)$.

Clause (1) If $\varphi \not\vdash x{\leq}y$, then $\Gamma_2(\varphi)$ disentails $x{\leq}y$ by forcing x to have a feature f_x which y must not have.

Clause (2) If $\varphi \not\vdash x{\sim}y$, then $\Gamma_2(\varphi)$ disentails $x{\sim}y$ by forcing x and y to have a common feature f_{xy} such that the subtrees of x and y at f_{xy} are incompatible.

Clauses (3) and (4) If $\varphi \not\vdash a(x)$, then $\Gamma_2(\varphi)$ disentails $a(x)$ for every sort a by forcing x to be consistent with two trees with distinct label. ■

DEFINITION (SATURATION) *Let φ be an* F*-closed constraint. By Lemma 7, $\Gamma_1(\varphi)$ is satisfiable such that we can define a saturation* $\mathrm{Sat}(\varphi)$ *of φ by* $\mathrm{Sat}(\varphi) {=_{def}} \Gamma_2(\Gamma_1(\varphi))$.

LEMMA 10 (SATURATION CHARACTERIZES SYNTACTIC ENTAILMENT) *Let φ be an* F*-closed constraint and μ an atomic constraint such that $V(\mu) \subseteq V(\varphi)$ and $F(\mu) \subseteq F(\varphi)$. Then $\varphi \not\vdash \mu$ implies* $\mathrm{Sat}(\varphi) \models \neg\mu$.

Proof: Let φ be an F-closed constraint and μ an atomic constraint such that $V(\mu) \subseteq V(\varphi)$ and $F(\mu) \subseteq F(\varphi)$. Suppose that $\varphi \not\vdash \mu$. Hence $\Gamma_1(\varphi) \not\vdash \mu$ by Property (P1) of Lemma 7. If μ is not a selection constraint then $\Gamma_2(\Gamma_1(\varphi)) \models \neg\mu$ by Lemma 9 and $V(\mu) \subseteq V(\varphi)$. Otherwise, let $\mu = x[f]y$ for some $x,y \in V(\varphi)$ and $f \in F(\varphi)$. Hence, $\Gamma_1(\varphi) \not\vdash v_{xf}{\leq}y$ or $\Gamma_1(\varphi) \not\vdash y{\leq}v_{xf}$ by Property (P2) of Lemma 7. By Lemma 9, either $\Gamma_2(\Gamma_1(\varphi)) \models \neg v_{xf}{\leq}y$ or $\Gamma_2(\Gamma_1(\varphi)) \models \neg y{\leq}v_{xf}$ holds, and hence again $\Gamma_2(\Gamma_1(\varphi)) \models \neg\mu$. ■

Proof of Lemma 1: We check that $\mathrm{Sat}(\varphi)$ has the three postulated properties. (1) The saturation formula $\mathrm{Sat}(\varphi)$ entails φ by construction. (2) Lemmas 7 and 8 prove that $\mathrm{Sat}(\varphi)$ is satisfiable. (3) By Lemma 10, $\mathrm{Sat}(\varphi)$ contradicts all atomic constraints μ with $V(\mu) \subseteq V(\varphi)$ and $F(\mu) \subseteq F(\varphi)$ that φ does not support syntactically.

10. Conclusion

We have presented the constraint system $FT_{\leq}$ of ordering constraints over feature trees. We have shown that the satisfiability problem of $FT_{\leq}$ and its entailment problem can be solved in cubic time and have given correct and complete algorithms for both. We have proved the independence property of $FT_{\leq}$, which implies that conjunctions of positive and negative ordering constraints $\varphi \wedge \neg\varphi_1 \wedge \ldots \wedge \neg\varphi_n$ can also be tested for satisfiability in cubic time. Finally, we have shown that our satisfiability test for positive $\mathrm{FT}_{\leq}$ constraints improves the known complexity of the satisfiability problem for weak subsumption constraints from $O(n^5)$ to $O(n^3)$.

Acknowledgments

We would like to thank Jochen Dörre, Gert Smolka, and Ralf Treinen for discussions on the topic of this paper. We thank Kartin Erk for having checked the final manuscript. We would also like to acknowledge the many helpful remarks of the referees. The research reported in this paper has been supported by the Esprit Working Group CCL II (EP 22457) and the Deutsche Forschungsgemeinschaft DFG through the SFB 378 at the Universität des Saarlandes.

Notes

1. The proof given in [11] assumes infinite sets of features and node labels. We conjecture (and this should not be too difficult to prove) that the first-order theory of FT is also completely axiomatizable for finite signatures.
2. A feature algebra is not an algebra since its features are interpreted as partial but not total functions

References

1. Hassan Aït-Kaci. An algebraic semantics approach to the effective resolution of type equations. *Theoretical Computer Science*, 45:293–351, 1986.
2. Hassan Aït-Kaci and R. Nasr. LOGIN: A logic programming language with built-in inheritance. *Journal on Lisp and Symbolic Computation*, 2:51–89, 1989.
3. Hassan Aït-Kaci and Roger Nasr. LOGIN: A logic programming language with built-in inheritance. *The Journal of Logic Programming*, 3(3):185–215, 1986.
4. Hassan Aït-Kaci and Andreas Podelski. Entailment and disentailment of order-sorted feature constraints. In Andrei Voronkov, editor, *Proceedings of the 4th International Conference on Logic Programming and Automated Reasoning*, volume 698 of *Lecture Notes in Artificial Intelligence*, pages 1–18. Springer-Verlag, Berlin, July 1993.
5. Hassan Aït-Kaci and Andreas Podelski. Towards a meaning of life. *The Journal of Logic Programming*, 16(3 – 4):195–234, July, August 1993.
6. Hassan Aït-Kaci, Andreas Podelski, and Gert Smolka. A feature-based constraint system for logic programming with entailment. *Theoretical Computer Science*, 122(1–2):263–283, January 1994.
7. Rolf Backofen. *Expressivity and Decidability of First-order Languages over Feature Trees*. Doctoral Dissertation. Universität des Saarlandes, Technische Fakultät, D–66041 Saarbrücken, 1994.
8. Rolf Backofen. Regular path expressions in feature logic. *Journal of Symbolic Computation*, 17:421–455, 1994.

9. Rolf Backofen. A complete axiomatization of a theory with feature and arity constraints. *The Journal of Logic Programming*, 24(1 – 2):37–71, 1995. Special Issue onComputational Linguistics and Logic Programming.
10. Rolf Backofen. Controlling functional uncertainty. In Wolfgang Wahlster, editor, *Proceedings of* 12^{th} *European Conference on Artificial Intelligence*, pages 557–561. John Wiley & Sons, Ltd, 1996.
11. Rolf Backofen and Gert Smolka. A complete and recursive feature theory. *Theoretical Computer Science*, 146(1–2):243–268, July 1995.
12. Rolf Backofen and Ralf Treinen. How to win a game with features. In Jean-Pierre Jouannaud, editor, *1st International Conference on Constraints in Computational Logics*, Lecture Notes in Computer Science, vol. 845, pages 320–335, München, Germany, September 1994. Springer-Verlag.
13. Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proceedings of the National Conference on Artificial Intelligence*, pages 34–37, August 1984.
14. Bob Carpenter. *The Logic of Typed Feature Structures - with Applications to Unification Grammars, Logic Programs and Constraint Resolution*. Number 32 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, England, 1992.
15. Witold Charatonik and Andreas Podelski. The independence property of a class of set constraints. In Eugene C. Freuder, editor, *Proceedings of the* 2^{nd} *International Conference on Principles and Practice of Constraint Programming*, volume 1118 of *Lecture Notes in Computer Science*, pages 76–90, 1996.
16. Witold Charatonik and Andreas Podelski. Set constraints with intersection. In *Proceedings of the* 12^{th} *IEEE Symposium on Logic in Computer Science*, pages 352–361, Warsaw, Poland, 1997. IEEE Computer Society Press.
17. Alain Colmerauer. Equations and inequations on finite and infinite trees. In ICOT, editor, *Proceedings of the* 2^{nd} *International Conference on Fifth Generation Computer Systems*, pages 85–99. Omsha Ltd., Tokyo and North-Holland, Amsterdam, 1984.
18. Martin Dietzfelbinger, Anna Karlin, Kurt Mehlhorn, Friedhelm Meyer Auf Der Heide, Hans Rohnert, and Robert E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. *SIAM Journal of Computing*, 23(4):738–761, August 1994.
19. Jochen Dörre. Feature-logic with weak subsumption constraints. In M. A. Rosner C. J. Rupp and R. L. Johnson, editors, *Constraints, Languages, and Computation*, chapter 7, pages 187–203. Academic Press, 1994.
20. Jochen Dörre. *Feature-Logik und Semiunifikation*. Number 128 in DISKI - Dissertationen zur Künstlichen Intelligenz. Infix Verlag, Sankt Augustin, July 1996. In German.
21. Jochen Dörre and William C. Rounds. On subsumption and semi-unification in feature algebras. *Journal of Symbolic Computation*, 13:441–461, 1992.
22. R. Helm, K. Marriott, and M. Odersky. Constraint-based query optimization for spatial databases. In 10^{th} *Annual IEEE Symposium on the Principles of Database Sytems*, pages 181–191, May 1991.
23. Mark Johnson. *Attribute-Value Logic and the Theory of Grammar*. Number 16 in CSLI Lecture Notes. Center for the Study of Language and Information, 1988.
24. Ronald M. Kaplan and Joan Bresnan. Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press, Cambridge, MA, 1982.
25. Robert T. Kasper and William C. Rounds. A logical semantics for feature structures. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, pages 257–265, 1986.
26. Martin Kay. Functional grammar. In C. Chiarello et al., editor, *Proceedings of the* 5^{th} *Annual Meeting of the Berkeley Linguistics Society*, pages 142–158, 1979.
27. J. Lassez and K. McAloon. Applications of a canonical form for generalized linear constraints. In *Proceedings of the* 5^{th} *International Conference on Fifth Generation Computer Systems*, pages 703–710, December 1988.
28. Kuniaki Mukai. Partially specified terms in logic programming for linguistic analysis. In *Proceedings of the* 6^{th} *International Conference on Fifth Generation Computer Systems*, Tokyo, Japan, 1988. ICOT.
29. Martin Müller. Ordering constraints over feature trees with ordered sorts. In P. Lopez, Suresh Manandhar, and Werner Nutt, editors, *Computational Logic and Natural Language Understanding*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, to appear. Available at `http://www.ps.uni-sb.de/~mmueller/papers/clnlp.ps.Z`.
30. Martin Müller and Joachim Niehren. Entailment for set constraints is not feasible. Technical report, Programming Systems Lab, Universität des Saarlandes, 1997. `http://www.ps.uni-sb.de/Papers/abstracts/inesInfeas.html`.

31. Martin Müller and Joachim Niehren. Ordering constraints over feature trees expressed in second-order monadic logic. In Tobias Nipkow, editor, *International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 196–210, Tsukuba, Japan, 1998. Springer-Verlag, Berlin.
32. Martin Müller, Joachim Niehren, and Andreas Podelski. Inclusion constraints over non-empty sets of trees. In Michel Bidoit and Max Dauchet, editors, *Proceedings of the Theory and Practice of Software Development*, volume 1214 of *Lecture Notes in Computer Science*, pages 345–356, Lille, France, April 1997. Springer-Verlag, Berlin.
33. Martin Müller, Joachim Niehren, and Ralf Treinen. The first-order theory of ordering constraints over feature trees. In *Proceedings of the 13th IEEE Symposium on Logic in Computer Science*, pages 432–443. IEEE Computer Society Press, 1998.
34. Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, 1990.
35. Bernhard Nebel and Gert Smolka. Representation and reasoning with attributive descriptions. In K.H. Bläsius, U.Hedtstück, and C.-R. Rollinger, editors, *Sorts and Types in Artificial Intelligence*, volume 418 of *Lecture Notes in Artificial Intelligence*, pages 112–139. Springer-Verlag, Berlin, 1990.
36. Joachim Niehren, Martin Müller, and Jean-Marc Talbot. Entailment of atomic set constraints is PSPACE-complete, December 1998. `www.ps.uni-sb.de/Papers/abstracts/atomic:98.html`.
37. Jens Palsberg. Efficient inference of object types. In *Proceedings of the 9th IEEE Symposium on Logic in Computer Science*, pages 186–185. IEEE Computer Society Press, 1994.
38. Andreas Podelski and Gert Smolka. Operational semantics of constraint logic programs with coroutining. In Leon Sterling, editor, *Proceedings of the 12th International Conference on Logic Programming*, pages 449–463, Kanagawa, Japan, 13–18 June 1995. The MIT Press, Cambridge, MA.
39. Carl Pollard and Ivan Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. Cambridge University Press, Cambridge, England, 1994.
40. Carl J. Pollard and Ivan A. Sag. *Information-based Syntax and Semantics, Vol. 1*. Number 13 in CSLI Lecture Notes. Center for the Study of Language and Information, Stanford University, 1987. Distributed by University of Chicago Press.
41. François Pottier. Simplifying subtyping constraints. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, pages 122–133. ACM Press, New York, May 1996.
42. William C. Rounds. Feature logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, pages 475–533. Elsevier Science Publishers B.V. (North Holland), 1997. Part 2: General Topics.
43. Steward Shieber. *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes No. 4. Center for the Study of Language and Information, 1986.
44. Steward Shieber. *Parsing and Type Inference for Natural and Computer Languages*. SRI International Technical Note 460, Stanford University, March 1989.
45. Steward Shieber, Hans Uszkoreit, Fernando Pereira, J. Alan Robinson, and M. Tyson. The formalism and implementation of PATR-II. In Joan Bresnan, editor, *Research on Interactive Acquisition and Use of Knowledge*. SRI International, Menlo Park, California, 1983.
46. Gert Smolka. Feature constraint logics for unification grammars. *The Journal of Logic Programming*, 12(1 – 2):51–87, 1992.
47. Gert Smolka. The Oz Programming Model. In Jan van Leeuwen, editor, *Computer Science Today*, volume 1000 of *Lecture Notes in Computer Science*, pages 324–343. Springer-Verlag, Berlin, 1995.
48. Gert Smolka and Ralf Treinen. Records for logic programming. *The Journal of Logic Programming*, 18(3):229–258, April 1994.
49. Ralf Treinen. Feature constraints with first-class features. In Andrzej M. Borzyszkowski and Stefan Sokołowski, editors, *International Symposium on Mathematical Foundations of Computer Science*, volume 711 of *Lecture Notes in Computer Science*, pages 734–743, Gdańsk, Poland, 30 August–3 September 1993. Springer-Verlag, Berlin.
50. Andreas Zeller and Gregor Snelting. Handling version sets through feature logic. In W. Schäfer and P. Botella, editors, *Proceedings of the 5th European Software Engineering Conference*, volume 989 of *Lecture Notes in Computer Science*, pages 191–204, Sitges, Spain, September 1995. Springer-Verlag, Berlin.
51. Andreas Zeller and Gregor Snelting. Unified versioning through feature logic. *ACM Transactions on Software Engineering and Methodology*, 6(4):398–441, October 1997.