Ordering Constraints over Feature Trees

Martin Müller[1], Joachim Niehren[1] and Andreas Podelski[2]

[1] Universität des Saarlandes, {mmueller,niehren}@ps.uni-sb.de
[2] Max-Planck-Institut für Informatik, podelski@mpi-sb.mpg.de
Saarbrücken, Germany

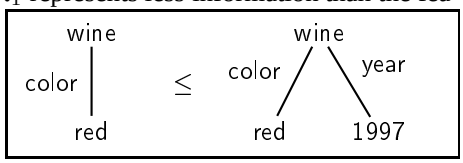
Abstract. Feature trees have been used to accommodate records in constraint programming and record like structures in computational linguistics. Feature trees model records, and feature constraints yield extensible and modular record descriptions. We introduce the constraint system $FT_{\leq}$ of ordering 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 a polynomial algorithm that decides the satisfiability of conjunctions of positive and negative information ordering constraints over feature trees. Our results include algorithms for the satisfiability problem and the entailment problem of $FT_{\leq}$ in time $O(n^3)$. We also show that $FT_{\leq}$ has the independence property and are thus able to handle negative conjuncts via entailment. Furthermore, we reduce the satisfiability problem of Dörre's weak-subsumption constraints to the satisfiability problem of $FT_{\leq}$. This improves 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 constraints have been used for describing records in constraint programming [2, 24, 23] and record like structures in computational linguistics [13, 12, 20, 18, 19]. Following [3, 5, 4] we consider feature constraints as predicate logic formulae that are interpreted in the structure of feature trees.

A feature tree is a possibly infinite tree with unordered labeled edges and with possibly labeled nodes. Edge labels are functional; i.e., the labels of the edges departing from the same node must be pairwise different. Under the view that feature trees represent symbolic information, the feature tree τ_1 represents less information than the feature tree τ_2 if τ_1 has fewer edges and node labels than τ_2. The relation $\leq$ that we define corresponds to the information ordering in precisely this sense. Algebraically, $\tau_1 \leq \tau_2$ if there is a *homomorphic embedding* from τ_1 to τ_2 (i.e., a mapping from nodes in τ_1 to nodes in τ_2 under which the node labeling is invariant). An example is given in the picture.



We introduce the constraint system $FT_{\leq}$ of information ordering constraints over feature trees. The system $FT_{\leq}$ is obtained by adding ordering constraints to the constraint

system FT [3]. The syntax of $FT_{\leq}$ constraints φ is defined by

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

where x and x' are variables and a is a label. The semantics of $FT_{\leq}$ is given by the interpretation over feature trees where the symbol $\leq$ is interpreted as information ordering on feature trees. The semantics of $x[a]y$ and $a(x)$ are defined as in FT. For instance, both trees depicted above are possible values for x in solutions of the constraint $\mathsf{wine}(x) \wedge x[\mathsf{color}]x' \wedge \mathsf{red}(x')$.

It is clear that $FT_{\leq}$ is more expressive than FT since the information ordering is antisymmetric (*i.e.*, $(x{\leq}x' \wedge x'{\leq}x) \leftrightarrow x{=}x'$ is valid). As we show in the paper, $FT_{\leq}$ is strictly more expressive than FT. For instance, no constraint in FT can be equivalent to $x{\leq}x'$. Also, we do not know of any formula over FT (even with existential quantifiers) equivalent to $\exists x\,(x_1{\leq}x \wedge x_2{\leq}x) \wedge \exists x\,(x_1{\leq}x \wedge x_3{\leq}x)$; this $FT_{\leq}$ formula expresses that x_1 is unifiable with both x_2 and x_3 (but does not imply unifiability of x_2 and x_3).

We show that the satisfiability problem of conjunctions of positive and negative $FT_{\leq}$ constraints $\varphi \wedge \neg\varphi_1 \wedge \ldots \wedge \neg\varphi_n$ is decidable in $O(n^3)$. This result includes a decision procedure for the entailment problem of the form $\varphi' \models \varphi$ since a formula $\varphi' \rightarrow \varphi$ is valid if and only if the formula $\varphi' \wedge \neg\varphi$ is unsatisfiable. To establish our result, we prove that $FT_{\leq}$ has the fundamental independence property (similar to its relatives RT [6], FT [3], and CFT [24]).

We reduce the satisfiability problem of Dörre's weak-subsumption constraints [7] over feature algebras linearly to the one in $FT_{\leq}$. Thereby, our algorithm improves on the best known satisfiability test for weak subsumption constraints which uses finite automata techniques and has an $O(n^5)$-complexity bound [7].

Plan of the Paper. Section 2 surveys related work. Section 3 defines $FT_{\leq}$. Section 4 presents the satisfiability test for $FT_{\leq}$ constraints. Section 8 contains the completeness proof. Section 5 presents the entailment test for $FT_{\leq}$ constraints, and proves the independence property of $FT_{\leq}$. Section 6 defines weak subsumption constraints and reduces their satisfiability problem to the one of $FT_{\leq}$ constraints. Section 7 shows that $FT_{\leq}$ is strictly more expressive than FT.

2 Related Work

Ines Constraints. In previous work [17], we have introduced the constraint system INES of inclusion constraints over non-empty sets of trees and a cubic satisfiability test. The satisfiability test for $FT_{\leq}$ is inspired by and subsumes the one for INES. However, the entailment problems for $FT_{\leq}$ and INES constraints are different. The entailment problem of INES constraints is coNP-hard [16]. Intuitively, the entailment problem of $FT_{\leq}$ is less expressive than the one of INES because an $FT_{\leq}$ constraint φ cannot uniquely describe a single feature tree (in absence of arity constraints); in contrast, INES constraints (which are inclusions between first-order terms with an implicit arity restriction) can uniquely describe a constructor tree as a singleton set. For instance, the INES constraint $x{\subseteq}a$ describes the singleton $\{a\}$. As a consequence, the entailment proposition $x{\subseteq}a \wedge a{\subseteq}y \models x{\subseteq}y$ holds in INES. No similar entailment phenomenon exists for $FT_{\leq}$.

Feature Constraints. The constraint system CFT [24] 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 [6] but provides finer-grained constraints. The system EF [25] extends CFT by feature constraints $x[y]z$, providing for first-class features. Complete axiomatizations for FT and CFT have been given in [5] and [4], respectively. The satisfiability of EF constraints is shown NP-hard in [25]. The system $FT_{\leq}(sort)$ extends $FT_{\leq}$ by allowing a partial order on labels [15].

Subsumption Constraints. Subsumption is an ordering on the domain of feature algebras. Subsumption constraints have been considered in the context of unification-based grammars to model coordination phenomena in natural language [9, 7, 21]. 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 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 [9], Dörre proposed weak subsumption as an decidable approximation of subsumption. As we show, the information ordering over feature trees (as investigated in this paper) coincides with the weak subsumption ordering interpreted over (the algebra of) feature trees.

Independent Constraint Systems. A constraint system has the fundamental *independence property* if negated conjuncts are independent from each other, or: its constraints cannot express disjunctions (we will give a formal definition later). Apart from the mentioned tree constraint systems RT, FT, CFT [6, 1, 24, 3], constraint systems with the independence property include linear equations over the real numbers [14], or infinite boolean algebras with positive constraints [10].

3 Syntax and Semantics of $\mathbf{FT}_{\leq}$

The constraint system $FT_{\leq}$ is defined by a set of constraints together with an interpretation over feature trees. We assume an infinite set of *variables* ranged over by x, y, z, and an infinite set $\mathcal{L}$ of *labels* ranged over by a, b.

Feature Trees. A *path* p is a finite sequence of labels. 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 prefix closed set of paths. A *feature tree* τ is a pair (D, L) consisting of a tree domain D and 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. The set of all feature trees is denoted by $\mathcal{F}$. A feature tree is called *finite* if its tree domain is finite, and *infinite* otherwise.

Syntax. An $FT_{\leq}$ *constraint* φ is defined by the following abstract syntax.

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

An $FT_{\leq}$ constraint is a conjunction of *basic constraints* which are either *inclusion constraints* $x{\leq}y$, *labeling constraints* $a(x)$, *selection constraints* $x[a]y$, or *compatibility constraints* $x{\sim}y$. Compatibility constraints are needed in our algorithm and can be expressed by first-order formulae over inclusion constraints (see Proposition 1). We identify $FT_{\leq}$ constraints φ up to associativity and commutativity of conjunction, *i.e.*, we view φ as a multiset of inclusion, 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 of label and variable occurrences in φ.

Semantics. We next define the structure $\mathcal{F}$ over feature trees in which we interpret $FT_{\leq}$ constraints. The signature of $\mathcal{F}$ contains the binary relation symbols $\leq$ and $\sim$ and for every label a a unary relation symbol $a()$ and a binary relation symbol $[a]$. In $\mathcal{F}$ these relation symbols are interpreted such:

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

Let Φ denote first-order formulae built from $FT_{\leq}$ constraints with the usual first order connectives. We call Φ *satisfiable* (valid) if Φ is satisfiable (valid) in the structure $\mathcal{F}$. We say that Φ *entails* Φ', written $\Phi \models \Phi'$, if $\Phi \to \Phi'$ is valid, and that Φ is *equivalent* to Φ' if $\Phi_1 \leftrightarrow \Phi_2$ is valid. We denote with $V(\Phi)$ the set of variables occurring free in Φ and with $L(\Phi)$ the set of labels occurring in Φ.

Proposition 1. *The formulae* $x{\sim}y$ *and* $\exists z\,(x{\leq}z \wedge y{\leq}z)$ *are equivalent in* $\mathcal{F}$.

Proof. Let σ be a variable assignment into $\mathcal{F}$ which also is a solution of 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 the variable assignment σ' defined by $\sigma'(z) = \tau$ and $\sigma'(x) = \sigma(x)$ for $x \neq z$ is a solution of $x{\leq}z \wedge y{\leq}z$. □

4 Satisfiability Test

We present a set of axioms valid for $FT_{\leq}$ and then interpret these axioms as an algorithm that solves the satisfiability problem of $FT_{\leq}$. The axioms and the algorithm are inspired by the ones for INES constraints presented in [17].

Table 1 contains five axiom schemes F1 - F5 that we regard as sets of axioms. The union of these sets of axioms is denoted by F, *i.e.*, $\mathsf{F} = \mathsf{F1} \cup \ldots \cup \mathsf{F5}$. For instance, an axiom scheme $x{\leq}x$ represents the infinite set of axioms obtained by instantiation of the meta variable x. An axiom is either a constraint φ, an implication between constraints $\varphi \to \varphi'$, or an implication $\varphi \to \mathit{false}$.

Proposition 2. *The structure* $\mathcal{F}$ *is a model of the axioms in* F.

F1	$x \leq x$ and $x \leq y \wedge y \leq z \rightarrow x \leq z$
F2	$x[a]x' \wedge x \leq y \wedge y[a]y' \rightarrow x' \leq y'$
F3	$x \leq y \rightarrow x \sim y$ and $x \leq y \wedge y \sim z \rightarrow x \sim z$ and $x \sim y \rightarrow y \sim x$
F4	$x[a]x' \wedge x \sim y \wedge y[a]y' \rightarrow x' \sim y'$
F5	$a(x) \wedge x \sim y \wedge b(y) \rightarrow \mathit{false}$ for $a \neq b$

Table 1. Satisfiability of $FT_{\leq}$ Constraints.

Proof. By a routine check. For illustration, we prove the statement for the second rule in F3, namely $x \leq y \wedge y \sim y' \rightarrow x \sim y'$. The following implications hold:

$$\begin{array}{lcll} x \leq y \wedge y \sim y' & \leftrightarrow & x \leq y \wedge \exists z (y \leq z \wedge y' \leq z) & \text{Proposition 1} \\ & \rightarrow & \exists z (x \leq z \wedge y \leq z) & \text{Transitivity} \\ & \leftrightarrow & x \sim y & \text{Proposition 1} \end{array}$$ □

The Algorithm F. The set of axioms F induces a fixed point algorithm F that, given an input constraint φ, iteratively adds logical consequences of $\mathsf{F} \cup \{\varphi\}$ to φ. (Observe that actually only constraints of the form $x \leq y$ and $x \sim y$ are derived). More precisely, in every step F inputs a constraint φ and terminates with *false* or outputs a constraint $\varphi \wedge \varphi'$. Termination with *false* takes place if there exists φ'' *in* φ such that $\varphi'' \rightarrow \mathit{false} \in \mathsf{F}$. Output of $\varphi \wedge \varphi'$ is possible if $\varphi' \in \mathsf{F}$ or there exists φ'' *in* φ with $\varphi'' \rightarrow \varphi' \in \mathsf{F}$.

Example 1. Inconsistency can be due to incompatible upper bounds. Consider:

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

We may add $x \sim z$ by F3.1, then $z \sim x$ via F3.3, then $y \sim x$ with F3.2, and finally terminate with *false* via F5.

Example 2. We need F4 for deriving the unsatisfiability of the constraint:

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

Algorithm F may add $x \sim y$ after several steps as shown in Example 1. Then it may proceed with $x \sim y'$ via F4 and terminate with *false* via F5.

Termination. The fixed point algorithm F terminates when reflexivity of inclusion $x \leq x$ (F1.1) is restricted to variables $x \in V(\varphi)$. Given a subset F of F, a constraint φ is called *F-closed* if algorithm F under this restriction and w.r.t. the axioms in F cannot proceed on φ. Note that *false* is not F-closed since it is not a constraint by definition.

Example 3. Our control takes care of termination in presence of cycles like $x[a]x$. For instance, the following constraint is F-closed.

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

In particular, $\mathsf{F2}$ and $\mathsf{F4}$ do not loop through the cycle $x[a]x$ infinitely often. This example also illustrates why the fixed point algorithm would not be terminating if based in the axiom $x[a]x' \wedge x{\leq}y \rightarrow \exists y'\,(y[a]y' \wedge x'{\leq}y')$.

Proposition 3. *If φ is a constraint with m variables then algorithm F with input φ terminates under the above control in at most $2 \cdot m^2$ steps.* □

Proof. Since F does not introduce new variables, it may add at most m^2 non-disjointness constraints $x{\sim}y$ and m^2 inclusions $x{\leq}y$. □

Proposition 4. *Every F-closed constraint φ is satisfiable over $FT_{\leq}$.*

Proof. See Section 8. □

Theorem 5. *The satisfiability of $FT_{\leq}$ constraints can be decided in time $O(n^3)$ (offline and online, see [11]) where n is the constraint size.*

Proof. Proposition 2 shows that φ is unsatisfiable if F started with φ terminates with *false*. Proposition 4 proves that φ is satisfiable if F started with φ terminates with a constraint. Since F terminates for all input constraints under the above control (Prop. 3) this yields a effective decision procedure. The main idea of the complexity proof is that one needs at most $O(n^2)$ steps (Prop. 3) each of which can be implemented in time $O(n)$. The implementation can be organized incrementally by exploiting that algorithm F leaves the order unspecified in which the axioms are applied. Hence, we obtain that off-line and on-line complexity are the same. The implementation details and the complexity proof are omitted here, since they are similar to those presented in [17]. □

5 Entailment, Independence, Negation

In this section, we give a cubic algorithm testing entailment $\varphi' \models \varphi$ between $FT_{\leq}$ constraints φ' and φ. We then prove the independence property of $FT_{\leq}$. Hence we can solve conjunctions of positive and negative $FT_{\leq}$-constraints $\varphi \wedge \neg\varphi_1 \wedge \ldots \neg\varphi_n$ in time $O(n^3)$. A *basic constraint* μ is a conjunction free constraint φ, *i.e.*, given by the following abstract syntax:

$$\mu ::= x{\leq}y \mid x{\sim}y \mid a(x) \mid x[a]y$$

The entailment $\varphi' \models \varphi$ is equivalent to the fact that the entailment $\varphi \models \mu$ holds for all basic constraints μ *in* φ.

Next we characterize entailment problems $\varphi' \models \mu$ syntactically. We say that a constraint φ *syntactically contains* μ, written $\varphi \vdash \mu$, if one of the following holds:

$$\begin{array}{ll}
\varphi \vdash a(x) & \text{if exists } x' \text{ such that } x' \leq x \wedge a(x') \in \varphi \\
\varphi \vdash x \leq y & \text{if } x \leq y \text{ in } \varphi \text{ or } x = y \\
\varphi \vdash x \sim y & \text{if } x \sim y \text{ in } \varphi \text{ or } x = y \\
\varphi \vdash x[a]y & \text{if exist } x', y' \text{ such that } x'[a]y' \text{ in } \varphi, \\
 & \quad \text{and } \varphi \vdash x \leq x', \varphi \vdash x' \leq x \text{ and } \varphi \vdash y \leq y', \varphi \vdash y' \leq y
\end{array}$$

We say that a first-order formula Φ syntactically contains μ, $\Phi \vdash \mu$, if $\Phi = \varphi \wedge \Phi'$ for some φ and Φ' such that $\varphi \vdash \mu$.

Lemma 6. *Given a* F*-closed constraint* φ*, we can compute a representation of* φ *in linear time that allows to test syntactic containment* $\varphi \vdash \mu$ *for all* μ *in time* $O(1)$.

Proof. Simple. □

It is easy to see that syntactic containment is semantically correct, *i.e.*, $\varphi \vdash \mu$ implies $\varphi \models \mu$. For deciding entailment, we have to show that our notion of syntactic containment is semantically complete, *i.e.*, if $\varphi \not\vdash \mu$ then $\varphi \not\models \mu$ (Proposition 13). The idea is to construct a satisfiable extension of φ (its saturation) which syntactically and simultaneously contradicts all μ not syntactically contained by φ (Lemma 12).

Saturation is defined in terms of two operators Γ_1 and Γ_2 on constraints. The operator Γ_2 is such that $\Gamma_2(\varphi)$ contradicts all μ of the form $x \sim y$, $x \leq y$, and $a(x)$ (*i.e.*, no selection constraints) which are not syntactically contained in φ (Lemma 10). The operator Γ_1 serves for contradicting selection constraints. For instance, consider $\varphi \models x[a]y$ where $\varphi = x \leq x \wedge y \leq y$. In this case, $\Gamma_1(\varphi)$ enforces the existence of the feature a in the denotation of x by adding to φ the constraint $x[a]v_{xa}$ for a fresh variable v_{xa}. Now $\Gamma_2(\Gamma_1(\varphi))$ is such that it contradicts either $y \leq v_{xa}$ or $v_{xa} \leq y$. (see Example 4). In this sense, Γ_1 is a "preprocessor" for Γ_2.

Definition 7. Let φ be a constraint, v_1 and v_2 distinct fresh variables, and l_1 and l_2 distinct labels. Furthermore, for every pair of variables $x, y \in V(\varphi)$, and label every label $a \in L(\varphi)$ let l_x and l_{xy} be fresh labels and v_{xa} a fresh variable. We define $\Gamma_1(\varphi)$ and $\Gamma_2(\varphi)$ in dependence of $v_1, v_2, l_x, l_{xy}, v_x$ as follows:

$$\begin{array}{rcll}
\Gamma_1(\varphi) & = & \varphi \wedge \bigwedge\{x[a]v_{xa} \mid x \in V(\varphi),\ a \in L(\varphi)\} & \\
\Gamma_2(\varphi) & = & \varphi \wedge \bigwedge\{x[l_x]v_x \wedge \neg\exists y'(y[l_x]y') \mid \varphi \not\vdash x \leq y,\ x, y \in V(\varphi)\} & (1) \\
 & & \quad \wedge \bigwedge\{x[l_{xy}]v_1 \wedge y[l_{xy}]v_2 \mid \varphi \not\vdash x \sim y,\ x, y \in V(\varphi)\} & (2) \\
 & & \quad \wedge \bigwedge\{x \sim v_1 \wedge x \sim v_2 \mid \text{for all labels } a:\ \varphi \not\vdash a(x),\ x \in V(\varphi)\} & (3)
\end{array}$$

Example 4. Consider the constraint $\varphi_0 = x[a]x \wedge y \leq x$ which is F-closed up to trivial constraints and which does not entail $x[a]y$. In order to contradict $x[a]y$ we compute the F-closure of $\Gamma_1(\varphi_0)$ which is $\Gamma_1'(\varphi_0) = x[a]x \wedge y \leq x \wedge x[a]v_{xa} \wedge y[a]v_{ya} \wedge v_{ya} \leq v_{xa} \wedge v_{xa} \leq x \wedge x \leq v_{xa} \wedge y \leq v_{xa}$ and observe that it does not $v_{xa} \leq y$. By definition of Γ_2, $\Gamma_2(\Gamma_1'(\varphi))$ contradicts $v_{xa} \leq y$. Hence, $\Gamma_2(\Gamma_1'(\varphi))$ also contradicts $x[a]y$.

Lemma 8. *Let φ be an F-closed (and hence satisfiable) constraint. Then $\Gamma_1(\varphi)$ is satisfiable and its F closure $\Gamma_1'(\mu)$ satisfies the following two properties for all basic constraints μ:*

1. *If $\varphi \not\vdash \mu$ and $V(\mu) \subseteq V(\varphi)$, then $\Gamma_1'(\varphi) \not\vdash \mu$.*
2. *If $\varphi \not\vdash x[a]y$ then $\Gamma_1'(\varphi) \not\vdash y \leq v_{xa}$ or $\Gamma_1'(\varphi) \not\vdash v_{xa} \leq y$.*

Proof. The F-closure $\Gamma_1'(\varphi)$ of $\Gamma_1(\varphi)$ has the following form up-to trivial constraints and symmetry of compatibility constraints.

$$\Gamma_1'(\varphi) = \Gamma_1(\varphi) \wedge \bigwedge\{v_{xa} \leq v_{ya} \mid \Phi \vdash x \leq y, a \in L(\varphi)\} \tag{4.1}$$
$$\wedge \bigwedge\{z \leq v_{xa} \mid \text{exists } y : \Phi \vdash z \leq y' \wedge y[a]y' \wedge y \leq x, \Phi \not\vdash x[a]\} \tag{4.2}$$
$$\wedge \bigwedge\{v_{xa} \leq z \mid \text{exist } y, y' : \Phi \vdash x \leq y \wedge y[a]y' \wedge y' \leq z, \Phi \not\vdash x[a]\} \tag{4.3}$$
$$\wedge \bigwedge\{v_{xa} \sim z \mid \text{exist } y, y' : \Phi \vdash x \leq y \wedge y[a]y' \wedge y' \sim z, \Phi \not\vdash x[a]\} \tag{4.4}$$
$$\wedge \bigwedge\{v_{xa} \sim z \mid \text{exist } y, y' : \Phi \vdash x \sim y \wedge y[a]y' \wedge y' \leq z, \Phi \not\vdash x[a]\} \tag{4.5}$$

(For instance note that $v_{xa} \leq x' \wedge x' \leq v_{xa}$ *in* $\Gamma_1'(\varphi)$ if $x[a]x'$ *in* φ by clauses (4.2, 4.3) and reflexivity). All constraints in $\Gamma_1'(\varphi)$ either belong to $\Gamma_1(\varphi)$ or a derived from it by axioms in F. The F-closedness of $\Gamma_1'(\varphi)$ can be proved by a somewhat tedious case distinction. The same holds for the two additional properties of $\Gamma_1'(\varphi)$ claimed. □

Lemma 9. *If φ is F-closed then $\Gamma_2(\varphi)$ is satisfiable.*

Proof. It is not difficult to show that the constraint part of $\Gamma_2(\varphi)$ is F-closed up to trivial constraints ($x \leq x$ and $x \sim x$) and symmetric compatibility constraints. The critical bit is to check that the negated selection constraints added in clause (1) of $\Gamma_2(\varphi)$ are consistent. Let $\neg\exists y'(y[l_x]y')$ *in* $\Gamma_2(\varphi)$. We must show that $\Gamma_2(\varphi) \not\models \exists y'(y[l_x]y')$. Assume the converse, $\Gamma_2(\varphi) \models \exists y'(y[l_x]y')$. Then, by Corollary 27 in Section 8, there exist z and z' such that $\Gamma_2'(\varphi) \vdash z \leq y \wedge z[l_x]z'$. By definition of $\Gamma_2(\varphi)$ we know that $z = x$. However, if $\Gamma_2'(\varphi) \vdash x \leq y$ and hence (by definition of Γ_2) $\varphi \vdash x \leq y$ holds, clause (1) does not apply. Thus $\neg\exists y'(y[l_x]y')$ cannot be contained in $\Gamma_2(\varphi)$, in contradiction to our assumption. □

Lemma 10. *Let φ be an $FT_{\leq}$-constraint and let μ be a basic constraint of the form $x \sim y$, $x \leq y$, or $a(x)$ (i.e., not a selection constraint). Then $\Gamma_2(\varphi) \models \neg\mu$ if and only if $\varphi \not\vdash \mu$.*

Proof. By inspection of the definition of $\Gamma_2(\varphi)$. Clause (1) contradicts entailment of $x \leq y$ by φ by forcing x to have a feature l_x which y must not have. Clause (2) contradicts $x \sim y$ by forcing x and y to have a common feature l_{xy} such that the subtrees of x and y at l_{xy} are incompatible. Clause (3) contradicts $a(x)$ for any label by forcing x to be unlabeled (*i.e.*, compatible with at least two trees with distinct label). □

Definition 11 Saturation. Let φ be an F-closed constraint and $\Gamma_1'(\varphi)$ the F-closure of $\Gamma_1(\varphi)$ which exists according to Lemma 8. The *saturation of* φ is the formula $\mathrm{Sat}(\varphi)$ given by $\mathrm{Sat}(\varphi) = \Gamma_2(\Gamma_1'(\varphi))$.

Lemma 12. *Let φ be an F-closed constraint For all μ such that $V(\mu) \subseteq V(\varphi)$, $\varphi \not\vdash \mu$ implies* $\mathrm{Sat}(\varphi) \models \neg\mu$.

Proof. Let $\Gamma_1'(\varphi)$ the F-closure of $\Gamma_1(\varphi)$ such that $\mathrm{Sat}(\varphi) = \Gamma_2(\Gamma_1'(\varphi))$. If $\varphi \not\vdash \mu$ then $\Gamma_1'(\varphi) \not\vdash \mu$ by Lemma 8.1. If μ is not a selection constraint, then $\Gamma_2(\Gamma_1'(\varphi)) \models \neg\mu$ by Lemma 10. Otherwise, let $\mu = x[a]y$. Hence, $\Gamma_1'(\varphi) \not\vdash v_{xa} \leq y$ or $\Gamma_1'(\varphi) \not\vdash y \leq v_{xa}$ by Lemma 8.2. By Lemma 10, either $\Gamma_2(\Gamma_1'(\varphi)) \models \neg v_{xa} \leq y$ or $\Gamma_2(\Gamma_1'(\varphi)) \models \neg y \leq v_{xa}$ holds. In both cases, $\Gamma_2(\Gamma_1'(\varphi)) \models \neg\mu$ follows. □

Proposition 13. *The notions of entailment and of syntactic containment coincide for basic constraints: If φ is F-closed and μ a basic constraint then $\varphi \models \mu$ iff $\varphi \vdash \mu$.*

Proof. We assume $\varphi \models \mu$ and show $\varphi \vdash \mu$. (The converse is correctness of syntactic containment.) If $V(\mu) \not\subseteq V(\varphi)$ then μ is of the form $x \leq x$ or $x \sim x$ such that $\varphi \vdash \mu$. Otherwise, $V(\mu) \subseteq V(\varphi)$. If $\varphi \models \mu$, then $\mathrm{Sat}(\varphi) \models \mu$ since $\mathrm{Sat}(\varphi)$ contains φ. Moreover, $\mathrm{Sat}(\varphi)$ is satisfiable (Lemmas 8 and 9) such that $\mathrm{Sat}(\varphi) \not\models \neg\mu$. Hence, $\varphi \vdash \mu$ by Lemma 12. □

Theorem 14 Entailment. *Entailment problems of the form $\varphi' \models \varphi$ can be tested in cubic time.*

Proof. Let n be the size of $\varphi' \wedge \varphi$. To decide $\varphi' \models \varphi$, first test whether φ' is satisfiable. By Theorem 5 this can be done by computing the F-closure $\tilde{\varphi}'$ of φ' in time $O(n^3)$. If this test fails then the entailment test is trivial. Otherwise, from Lemma 12 we obtain $\tilde{\varphi}' \not\models \mu$ if $\varphi \not\vdash \mu$, and hence that $\tilde{\varphi}' \models \varphi$ iff $\tilde{\varphi}' \vdash \mu$ for all μ *in* φ. There are $O(n)$ such μ and $\tilde{\varphi}'$ is of size $O(n^2)$, hence, by Lemma 6, this is decidable in time $O(n)$. The overall complexity sums up to $O(n^3)$. □

Theorem 15 Independence. *The constraint system $FT_{\leq}$ has the independence property; i.e., for every $n \geq 1$ and constraints $\varphi, \varphi_1, \ldots, \varphi_n$:*

$$\text{if } \varphi \models \vee_{i=1}^{n} \varphi_i \text{ then } \varphi \models \varphi_i \text{ for some } i \in \{1, \ldots, n\}.$$

Proof. Assume $\varphi \models \vee_{i=1}^{n} \varphi_i$. If φ is unsatisfiable we are done. Also, if $\varphi \wedge \varphi_i$ is non-satisfiable for some j, then $\varphi \models \vee_{i=1}^{n} \varphi_i$ iff $\varphi \models \vee_{i=1, i \neq j}^{n} \varphi_i$ is. Now let φ and $\varphi \wedge \varphi_i$ be satisfiable for all i and let φ be F-closed (wlog. by Prop. 2). If there exists i with $\varphi \vdash \mu$ for all μ syntactically contained by φ_i, then $\varphi \models \varphi_i$ and we are done. Otherwise, for all i there exists μ_i such that $\varphi \not\vdash \mu_i$. Lemma 12 yields $\mathrm{Sat}(\varphi) \models \wedge_{i=1}^{n} \neg\varphi_i$. Since $\mathrm{Sat}(\varphi)$ is satisfiable (Lemma 8) and entails φ, this contradicts our assumption that $\varphi \models \vee_{i=1}^{n} \varphi_i$. □

Corollary 16 Negation. *The satisfiability of conjunctions of positive and negative $FT_{\leq}$ constraints $\varphi \wedge \neg\varphi_1 \wedge \ldots \wedge \neg\varphi_k$ can be tested in time $O(n^3)$ where n is the size of the given conjunction.*

Proof. If φ is non-satisfiable then $\varphi \wedge (\wedge_{i=1}^{n} \neg\varphi_i)$ is trivially non-satisfiable. By Proposition 5, satisfiability of φ is decidable in time $O(n^3)$. Now assume φ to be satisfiable. By the Independence Theorem 15, $\varphi \wedge (\wedge_{i=1}^{n} \neg\varphi_i)$ is nonsatisfiable if and only if $\varphi \models \varphi_i$ for some i. By Lemma 12 this is equivalent to the existence of i such that for all μ if $\varphi_i \vdash \mu$ then $\varphi \vdash \mu$. Overall, there are $O(n^2)$ candidates μ to be tested for syntactic containment and $O(n)$ possible φ_i. By Lemma 6, $\varphi \vdash \mu$ can be tested in time $O(1)$ such that the total complexity sums up to time $O(n^3)$. □

6 Weak Subsumption Constraints

We next introduce weak subsumption constraints that are used in computational linguistics [7]. We show that their satisfiability problem is subsumed by the one for $FT_{\leq}$.

Syntax. We assume given a set $\mathcal{C}$ of *constants* c and a set $\mathcal{D}$ of features d. We consider the set of labels $\mathcal{L} = \mathcal{C} \cup \mathcal{D}$. A weak subsumption constraint η is a $FT_{\leq}$ constraint of the following form.

$$\eta \ ::= \ c(x) \ \mid \ x[d]y \ \mid \ x{\leq}y \ \mid \ x{\sim}y \ \mid \ \eta \wedge \eta'$$

Note that compatibility constraints do not occur in [7]. We add them here to simplify our comparison.

Semantics. We interpret weak subsumption constraints over the whole class of feature algebras with the induced weak subsumption ordering, which we will define below.

A *feature algebra* $\mathcal{A}$ over $\mathcal{C}$ and $\mathcal{D}$ consists of a set $\mathsf{dom}^{\mathcal{A}}$ that is called the *domain* of $\mathcal{A}$, a unary relation $c()^{\mathcal{A}}$ on $\mathsf{dom}^{\mathcal{A}}$ for every constant $c \in \mathcal{C}$, and a binary relation $[d]^{\mathcal{A}}$ on $\mathsf{dom}^{\mathcal{A}}$ for every feature $d \in \mathcal{D}$, which satisfy the following properties for all $\alpha, \alpha', \alpha'' \in \mathsf{dom}^{\mathcal{A}}$, constants $c, c_1, c_2 \in \mathcal{C}$, and features $d \in \mathcal{D}$:

1. if $\alpha[d]^{\mathcal{A}}\alpha'$ and $\alpha[d]^{\mathcal{A}}\alpha''$ then $\alpha' = \alpha''$
2. if $c_1(\alpha)^{\mathcal{A}}$ and $c_2(\alpha)^{\mathcal{A}}$ then $c_1 = c_2$

In the literature [22, 7] a slightly different notion of *feature algebras with constants* has been considered. We will give a formal comparison between the two notions at the end of the present section.

Proposition 17. *The structure $\mathcal{F}$ over $\mathcal{L}$ is a feature algebra over $\mathcal{C}$ and $\mathcal{D}$.*

Proof. The above properties follow from the axioms in F and the antisymmetry of the information ordering in $FT_{\leq}$ ($x{\leq}y \wedge y{\leq}x \rightarrow x{=}y$). □

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 Δ on the domain of $\mathcal{A}$ that satisfies the following properties for all elements α_1, α_2, α_1', α_2' of $\mathcal{A}$'s domain:

1. if $\alpha_1 \Delta \alpha_2$, $c_1(\alpha_1)^{\mathcal{A}}$, and $c_2(\alpha_2)^{\mathcal{A}}$ then $c_1 = c_2$
2. if $\alpha_1 \Delta \alpha_2$, $\alpha_1[d]^{\mathcal{A}}\alpha_1'$, and $\alpha_2[d]^{\mathcal{A}}\alpha_2'$ then $\alpha_1' \Delta \alpha_2'$

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 $\sim^{\mathcal{A}}$:

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

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

Proposition 18 Dörre [8]. *The structure $\mathcal{F}$ coincides with the structure induced by the feature algebra defined by $\mathcal{F}$.*

Proof. It is sufficient to prove that the weak subsumption relation of the feature algebra defined by $\mathcal{F}$ coincides with the information ordering on $\mathcal{F}$. The proof in the case for feature algebras with constants can be found in [8] on page 24 (Satz 6 and Satz 7). There the algebra of feature trees has been called algebra of path functions. A direct proof (additional 5 lines) is omitted for lack of space. □

Theorem 19. *A weak subsumption constraint* η *is satisfiable* (*over* $\mathcal{F}$) *if and only if* η *is satisfiable over the structure induced by some feature algebra* $\mathcal{A}$.

Proof. If η is satisfiable then it is satisfiable over the structure induced by the feature algebra defined by $\mathcal{F}$. Conversely, every structure induced by a feature algebra is a model of the axioms in F. Thus, if η is satisfiable over one such structure then it is equivalent to an F-closed constraint (and not *false*) and hence satisfiable over $\mathcal{F}$. □

Alternative Notions of Feature Algebras. In the literature [22, 7] a restricted notion of feature algebra has been considered that we call *feature algebra with constants* in the sequel. The focus on feature algebras with constants leads to a restricted satisfiability problem. This shows that the presented results properly extend the results in [7].

A *feature algebra with constants* is a feature algebra with the additional property that

$$\text{if } c(\alpha)^{\mathcal{A}} \text{ then not } \alpha[d]^{\mathcal{A}}\alpha' \tag{1}$$

In order to handle the new property we consider the following mapping of weak subsumption constraints over $\mathcal{C}$ and $\mathcal{D}$ to weak subsumption constraints over $\mathcal{C} \cup \{|\mathsf{label}|\}$ and $\mathcal{D}$ where $|\mathsf{label}|$ is a new constant not contained in $\mathcal{C}$.

$$[\![c(x)]\!] = \exists y\,(x[|\mathsf{label}|]y \wedge c(y)) \qquad [\![x[d]y]\!] = x[d]y$$
$$[\![x{\leq}y]\!] = x{\leq}y \qquad [\![x{\sim}y]\!] = x{\sim}y \qquad [\![\eta \wedge \eta']\!] = [\![\eta]\!] \wedge [\![\eta']\!]$$

Proposition 20. *A constraint* η *is satisfiable in some feature algebra if and only if* $[\![\eta]\!]$ *is satisfiable in some feature algebra with constants.*

Proof. If $[\![\eta]\!]$ is satisfiable over a feature algebra $\mathcal{A}$ with constants $\mathcal{C}$ and features $\mathcal{D} \cup \{|\mathsf{label}|\}$ then η is satisfiable over the feature algebra $\mathcal{F}$ with labels $\mathcal{C} \cup \mathcal{D}$. Given a solution σ' of $[\![\eta]\!]$ over $\mathcal{A}$ a solution σ of η over $\mathcal{F}$ can be defined as follows:

$$D_{\sigma(x)} = \{p \mid \text{ exists } \alpha \text{ in domain of } \mathcal{A}\text{: } \sigma'(x)[p]^{\mathcal{A}}\alpha \text{ and } p \in \mathcal{D}^*\}$$
$$L_{\sigma(x)} = \{(p, c) \mid \text{ exists } \alpha \text{ in domain of } \mathcal{A} : \sigma'(x)[p|\mathsf{label}|]^{\mathcal{A}}\alpha \text{ and } c(\alpha)^{\mathcal{A}}\}$$

Conversely, let η be satisfiable in a feature algebra $\mathcal{A}$. Then η is satisfiable in $\mathcal{F}$ by Theorem 19. We consider the following feature algebra with constants $\mathcal{F}^*$ and show that $[\![\eta]\!]$ is satisfiable over $\mathcal{F}^*$. The constants and features of $\mathcal{F}^*$ are $\mathcal{C}$ and $\mathcal{D} \cup \{|\mathsf{label}|\}$, respectively. The domain of $\mathcal{F}^*$ contains all feature trees τ without labeled internal nodes where a *labeled internal node* of τ is a path p such that $p \in D_\tau$, exists c with $(p, c) \in L_\tau$, but not exists d with $pd \in D_\tau$. The selection and labeling relations of $\mathcal{F}^*$ are those of $FT_{\leq}$ restricted to trees without internal labels. Obviously, $\mathcal{F}^*$ satisfies all three axioms of a feature algebra with constants. Now let σ be an $\mathcal{A}$-solution of η. Then the variable assignment σ' mapping x on $\sigma'(x)$ as given below is an $\mathcal{F}^*$-solution of $[\![\eta]\!]$.

$$D_{\sigma'(x)} = D_{\sigma(x)} \cup \{p|\mathsf{label}| \mid \text{exists } a \in \mathcal{L} : (p, a) \in L_{\sigma(x)}\}$$
$$L_{\sigma'(x)} = \{(p|\mathsf{label}|, a) \mid (p, a) \in L_{\sigma(x)}\}$$ □

7 Expressiveness

We show that $FT_{\leq}$ is strictly more expressive than FT but that $FT_{\leq}$ cannot express an arity constraint. An FT constraint η is of the form $x{=}y$, $a(x)$, $x[a]y$, or $\eta \wedge \eta'$, and an arity constraint of the form $x\{a_1,\ldots,a_n\}$. An arity constraint $x\{a_1,\ldots,a_n\}$ holds if x denotes a tree with subtrees at exactly a_1 through a_n.

Proposition 21. *There is no $FT_{\leq}$ constraint which expresses that a variable x denotes the empty feature tree, i.e., if $a \neq b$ then there is no constraint equivalent to*

$$x\{\} \wedge \exists y \exists z\,(x{\leq}y \wedge x{\leq}z \wedge a(y) \wedge b(z))$$

Proof. If φ were such a $FT_{\leq}$ constraint, then φ as well as its finite F-closure would entail $x{\leq}y$ for all variables y. This contradicts Proposition 13 for all those y such that $y \notin \mathcal{V}(\varphi)$ and $x \neq y$ (because if $\varphi \vdash x{\leq}y$ then $x = y$ or $x,y \in \mathcal{V}(\varphi)$). Such a variable y exists since $V(\varphi)$ is finite. □

Lemma 22. *Let η be an FT constraint. Then $\eta \models x{\leq}y$ if and only if $\eta \models y{\leq}x$.*

Proof. The FT constraint η is equivalent to the $FT_{\leq}$ constraint φ obtained from η by replacing all equalities $x{=}y$ by inequalities $x{\leq}y \wedge y{\leq}x$. Hence, $x{\leq}y$ *in* φ iff $y{\leq}x$ *in* φ, and since algorithm F preserves this invariant it also holds for the F-closure of φ. The claim follows from Proposition 13. □

Proposition 23. *If $x \neq y$ then there is no FT constraint η equivalent to $x{\leq}y$.*

Proof. This follows immediately from Lemma 22 and Proposition 13. □

8 Completeness of the Satisfiability Test

Proposition 4. Every F-closed constraint φ is satisfiable over $FT_{\leq}$.

The proof is based on the notion of path reachability and covers the rest of the section. We proceed as follows. We first define path reachability, then give two Lemmas, and finally compose the proof of Proposition 4 from these Lemmas.

For all paths p and constraint φ, we define a binary relation $\overset{\varphi}{\rightsquigarrow}_p$, where $x \overset{\varphi}{\rightsquigarrow}_p y$ reads as "y is reachable from x over path p in φ":

$$\begin{array}{ll} x \overset{\varphi}{\rightsquigarrow}_{\varepsilon} y & \text{if } y{\leq}x \text{ in } \varphi \\ x \overset{\varphi}{\rightsquigarrow}_{a} y & \text{if } x[a]y \text{ in } \varphi, \\ x \overset{\varphi}{\rightsquigarrow}_{pq} y & \text{if } x \overset{\varphi}{\rightsquigarrow}_{p} z \text{ and } z \overset{\varphi}{\rightsquigarrow}_{q} y. \end{array}$$

Define relationships $x \overset{\varphi}{\rightsquigarrow}_p a$ meaning that "a can be reached from x over path p in φ":

$$x \overset{\varphi}{\rightsquigarrow}_p a \text{ if } x \overset{\varphi}{\rightsquigarrow}_p y \text{ and } a(y) \text{ in } \varphi,$$

For example, if φ is the constraint $x{\leq}y \wedge a(y) \wedge x[a]u \wedge x[b]z \wedge z[a]x \wedge b(z)$ then the following reachability propositions hold: $y \overset{\varphi}{\rightsquigarrow}_{\varepsilon} x$, $x \overset{\varphi}{\rightsquigarrow}_{b} z$, $x \overset{\varphi}{\rightsquigarrow}_{ba} y$, $x \overset{\varphi}{\rightsquigarrow}_{ba} x$, *etc.*, as well as $x \overset{\varphi}{\rightsquigarrow}_{\varepsilon} a$, $x \overset{\varphi}{\rightsquigarrow}_{b} b$, $x \overset{\varphi}{\rightsquigarrow}_{ba} a$, *etc.*

Definition 24 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 $x \overset{\varphi}{\rightsquigarrow}_{p} a$, $x{\leq}x$, and $x \overset{\varphi}{\rightsquigarrow}_{p} b$ then $a = b$.
2. If $x \overset{\varphi}{\rightsquigarrow}_{p} a$, $x{\sim}y$, and $y \overset{\varphi}{\rightsquigarrow}_{p} b$ then $a = b$.

Lemma 25. *Every* F1-F2*-closed and path consistent constraint is satisfiable.*

Proof. Let φ be F1-F2-closed and path consistent. We define the variable assignment min_{φ} into feature trees as follows:

$$D_{\mathsf{min}_{\varphi}(x)} = \{p \mid x \overset{\varphi}{\rightsquigarrow}_{p} y\} \qquad \text{and} \qquad L_{\mathsf{min}_{\varphi}(x)} = \{(p, a) \mid x \overset{\varphi}{\rightsquigarrow}_{p} a\}$$

The path consistency of φ condition 1 implies that $L_{\mathsf{min}_{\varphi}(x)}$ is a partial function. Thus $\mathsf{min}_{\varphi}(x)$ is a feature tree. We now verify that min_{φ} is a solution of φ.

- Let $x{\leq}y$ *in* φ. For all x', if $y \overset{\varphi}{\rightsquigarrow}_{p} x'$ then $x \overset{\varphi}{\rightsquigarrow}_{p} x'$ by the definition of path reachability. Thus, $D_{\mathsf{min}_{\varphi}(y)} \subseteq D_{\mathsf{min}_{\varphi}(x)}$. For all a if $y \overset{\varphi}{\rightsquigarrow}_{p} a$ then $x \overset{\varphi}{\rightsquigarrow}_{p} a$ by the definition of path reachability. Thus, $L_{\mathsf{min}_{\varphi}(y)} \subseteq L_{\mathsf{min}_{\varphi}(x)}$, *i.e.*, $\mathsf{min}_{\varphi}(y){\leq}\mathsf{min}_{\varphi}(x)$.
- Consider $x[a]y$ *in* φ. We have to prove for all p, z, and b the equivalences

  $$x \overset{\varphi}{\rightsquigarrow}_{ap} z \;\text{ iff }\; y \overset{\varphi}{\rightsquigarrow}_{p} z \qquad \text{and} \qquad x \overset{\varphi}{\rightsquigarrow}_{ap} b \;\text{ iff }\; y \overset{\varphi}{\rightsquigarrow}_{p} b$$

  The first equivalence is equivalent to $D_{\mathsf{min}_{\varphi}(y)} = \{p \mid ap \in D_{\mathsf{min}_{\varphi}(x)}\}$ and the second one to $L_{\mathsf{min}_{\varphi}(y)} = \{(p, b) \mid (ap, b) \in L_{\mathsf{min}_{\varphi}(x)}\}$. We start proving the first equivalence. If $y \overset{\varphi}{\rightsquigarrow}_{p} z$ then $x \overset{\varphi}{\rightsquigarrow}_{ap} z$ since $x[a]y$ *in* φ. Suppose $x \overset{\varphi}{\rightsquigarrow}_{ap} z$. By definition of path reachability there exists x' and y' such that

  $$x \overset{\varphi}{\rightsquigarrow}_{\varepsilon} x', \qquad x'[a]y', \qquad y' \overset{\varphi}{\rightsquigarrow}_{p} z.$$

  The F1-closedness of φ and $x \overset{\varphi}{\rightsquigarrow}_{\varepsilon} x'$ imply $x{\leq}x'$ *in* φ. The F2-closedness ensures $y{\leq}y'$ *in* φ such that $y \overset{\varphi}{\rightsquigarrow}_{p} z$ holds. We now prove the second equivalence above. If $x \overset{\varphi}{\rightsquigarrow}_{ap} b$ then there exists z such that $x \overset{\varphi}{\rightsquigarrow}_{ap} z$ and $b(z)$. The first equivalence implies $y \overset{\varphi}{\rightsquigarrow}_{p} z$ and thus $y \overset{\varphi}{\rightsquigarrow}_{p} b$. The converse is simple.
- Let $a(x)$ in φ. Reflexivity (F1.1-closedness) implies $x{\leq}x$ *in* φ. Thus $x \overset{\varphi}{\rightsquigarrow}_{\varepsilon} a$ such that $(\varepsilon, a) \in L_x$.
- 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 partial function. If $(p, a) \in L_{\mathsf{min}_{\varphi}(x)}$ and $(p, b) \in L_{\mathsf{min}_{\varphi}(y)}$ then $x \overset{\varphi}{\rightsquigarrow}_{p} a$ and $y \overset{\varphi}{\rightsquigarrow}_{p} b$. The path consistency of φ condition 2 implies $a = b$. □

Lemma 26. *Every* F3, F4, F5*-closed constraint is path consistent.*

Proof. Let φ be F3, F4, F5-closed. Condition 1 of Definition 24 follows from condition 2 of Definition 24 and F3.1-closedness. The proof of condition 2 is by induction on paths p. We assume x, y, a, and b such that $x \overset{\varphi}{\rightsquigarrow}_p a$, $x{\sim}y$ *in* φ, and $x \overset{\varphi}{\rightsquigarrow}_p b$. If $p = \varepsilon$, then there exist $n, m \geq 0$, $x_1, \ldots, x_n$, $y_1, \ldots y_m$ such that:

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

F3-closedness implies $x_n{\sim}y_m$ *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 $p = a'q$, then there exists there exist x', y', $\tilde{x}$, $\tilde{y}$ with:

$$x \overset{\varphi}{\rightsquigarrow}_\varepsilon x', \quad x'[a']\tilde{x} \text{ in } \varphi\,, \quad \tilde{x} \overset{\varphi}{\rightsquigarrow}_p a\,,$$
$$y \overset{\varphi}{\rightsquigarrow}_\varepsilon y', \quad y'[a']\tilde{y} \text{ in } \varphi\,, \quad \tilde{y} \overset{\varphi}{\rightsquigarrow}_p b\,.$$

Since $x{\sim}x'$ *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 $a{=}b$ holds by induction hypothesis. □

Proof of Proposition 4. If φ is F-closed then φ is path consistent by Lemma 26 and thus satisfiable by Lemma 25. □

Corollary 27. *Let* φ *be an* F*-closed constraint. Then* $\varphi \models \exists y(x[a]y)$ *if and only if there are variables* x' *and* y' *such that* $\varphi \vdash x'[a]y'$ *and* $\varphi \vdash x'{\leq}x$.

Proof. Assume $\varphi \not\vdash x'[a]y' \wedge x'{\leq}x$. Then it holds for the minimal solution min_φ of an F-closed constraint that $a \notin L_{\mathsf{min}_\varphi(y)}$. Hence $\varphi \not\models \exists y(x[a]y)$. □.

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

References

1. H. Aït-Kaci and A. Podelski. Entailment and Disentailment of Order-Sorted Feature Constraints. In A. Voronkov, editor, *4th International Conference on Logic Programming and Automated Reasoning*, LNAI 698, pp. 1–18. Springer, 1993.
2. H. Aït-Kaci and A. Podelski. Towards a Meaning of Life. *The Journal of Logic Programming*, 16(3 and 4):195–234, July, Aug. 1993.
3. H. Aït-Kaci, A. Podelski, and G. Smolka. A feature-based constraint system for logic programming with entailment. *Theoretical Computer Science*, 122(1–2):263–283, Jan. 1994.

4. R. Backofen. A Complete Axiomatization of a Theory with Feature and Arity Constraints. *The Journal of Logic Programming*, 1995. Special Issue on Computational Linguistics and Logic Programming.
5. R. Backofen and G. Smolka. A complete and recursive feature theory. *Theoretical Computer Science*, 146(1–2):243–268, July 1995.
6. A. Colmerauer. Equations and Inequations on Finite and Infinite Trees. In *2nd Future Generation Computer Systems*, pages 85–99, 1984.
7. J. Dörre. Feature-Logic with Weak Subsumption Constraints. In *Constraints, Languages, and Computation*, chapter 7, pages 187–203. Academic Press, 1994.
8. J. Dörre. Feature-Logik und Semiunifikation. Dissertationen zur Künstlichen Intelligenz, Band 128. Infix-Verlag, St. Augustin, 1996.
9. J. Dörre and W. C. Rounds. On Subsumption and Semiunification in Feature Algebras. In *5th IEEE Symposium on Logic in Computer Science*, pages 300–310. IEEE Computer Science Press, 1990.
10. R. Helm, K. Marriott, and M. Odersky. Constraint-based Query Optimization for Spatial Databases. In *10th Annual IEEE Symposium on the Principles of Database Systems*, pages 181–191, May 1991.
11. J. Jaffar and M. J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–582, May-July 1994.
12. R. M. Kaplan and J. Bresnan. Lexical-Functional Grammar: A Formal System for Grammatical Representation. pages 173–381. MIT Press, Cambridge, MA, 1982.
13. M. Kay. Functional Grammar. In C. Chiarello et al., editor, *Proc. of the 5th Annual Meeting of the Berkeley Linguistics Society*, pages 142–158, 1979.
14. J. Lassez and K. McAloon. Applications of a Canonical Form for Generalized Linear Constraints. In *5th Future Generation Computer Systems*, pages 703–710, Dec. 1988.
15. M. Müller. Ordering Constraints over Feature Trees with Ordered Sorts. In P. Lopez, S. Manandhar, and W. Nutt, eds., *Computational Logic and Natural Language Understanding*, Lecture Notes in Artificial Intelligence, to appear, 1997.
16. M. Müller and J. Niehren. Entailment for Set Constraints is not Feasible. Technical report, Programming Systems Lab, Universität des Saarlandes, 1997. Available at `http://www.ps.uni-sb.de/~mmueller/papers/conp97.html`.
17. M. Müller, J. Niehren, and A. Podelski. Inclusion Constraints over Non-Empty Sets of Trees. In International Joint Conference on Theory and Practice of Software Development (TAPSOFT), LNCS, Springer, 1997.
18. C. Pollard and I. Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. Cambridge University Press, Cambridge, England, 1994.
19. W. C. Rounds. Feature Logics. In J. v. Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier Science Publishers B.V. (North Holland), 1997.
20. S. Shieber. *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes No. 4. Center for the Study of Language and Information, 1986.
21. S. Shieber. *Parsing and Type Inference for Natural and Computer Languages*. SRI Internationax[l Technical Note 460, Stanford University, Mar. 1989.
22. G. Smolka. Feature constraint logics for unification grammars. *Journal of Logic Programming*, 12:51–87, 1992.
23. G. Smolka. The Oz Programming Model. In J. van Leeuwen, editor, *Computer Science Today*, LNCS, vol. 1000, pages 324–343. Springer-Verlag, Berlin, Germany, 1995.
24. G. Smolka and R. Treinen. Records for Logic Programming. *The Journal of Logic Programming*, 18(3):229–258, Apr. 1994.
25. R. Treinen. Feature constraints with first-class features. *Mathematical Foundations of Computer Science*, LNCS, vol. 711, pages 734–743, Springer-Verlag, 1993.