Constraints, 5, 1–22 (1997)


# 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 January 1, 2000; Revised January 1, 2000*

**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 $O(n^3)$.

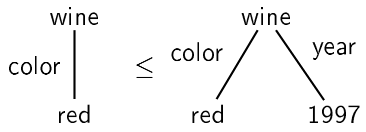
**Keywords:** feature constraints, tree orderings, weak subsumption, satisfiability, entailment, complexity

## 1. Introduction

Feature logic is a formalism to describe sets of objects by the values of their attributes or *features*. It has its origin in the three areas of knowledge representation, with *concept descriptions*, *frames* or $\psi$-terms [9, 30, 31, 1], natural language processing, in particular approaches based on *unification grammars* [22, 20, 38, 36, 33, 35], and constraint programming languages with record structures [3, 5, 24, 40]. An interesting recent application lies in software configuration management, where feature logic is used to denote software versions and to deduce their mutual consistency [43, 44].

The first mathematical treatment of record descriptions is the formalisms of $\psi$-*terms* [1], also named *feature structures* [34]: the attributes, having a functional meaning, are called features. For a thorough introduction to feature structures, see [10] or [35]. Logical reformulations of feature structures lead to the notion of *feature logic* [21, 19, 39]. In the logical view, a feature structure describing a set of record-like objects is generalized to a *feature constraint* expressing a relation (and unification of feature structures becomes constraint solving). In the first formalizations of feature logics, the constraints were interpreted in a class of interpretation domains. It is sometimes, e.g., for programming purposes, convenient to have one single domain of data. These data, which one can think of as nested records, are formalized through *feature trees* in [6, 8, 42, 41, 7].

A feature tree is a tree with labeled edges. The labels of the edges departing from the same node are pairwise different (in the special case of standard trees, the labels are just consecutive numbers). Branches may be infinite in order to model records with cyclic dependencies. Clearly, the edges correspond to record field selectors and the node labels to record field contents. The feature constraints in the constraint systems FT [6, 8] are conjunctions of two kinds of atomic formulas, besides equalities between variables:




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

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

For instance, both trees depicted above are possible values for $x$ in solutions of the constraint $\mathsf{wine}(x) \wedge x[\mathsf{color}]x' \wedge \mathsf{red}(x')$, but only the second one also can satisfy $x[\mathsf{year}]x''$. In contrast with the specification of standard trees using fixed-arity function symbols, e.g., by $x = \mathsf{wine}(red, 1997, \_, \ldots, \_)$, the description via feature constraints is modular and extensible (individual descriptions of the node label and each subtree; extension is done by incremental addition of atomic conjuncts).

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 compuational linguistics; see below), we may need a weaker constraint on $x$ and $y$. Under the view that feature trees represent symbolic information, 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 $\mathrm{FT}_{\leq}$ by adding ordering constraints $x{\leq}y$ as atomic formulas to the constraint system FT of [6]. The abstract syntax of $\mathrm{FT}_{\leq}$ constraints $\varphi$ is defined as follows:

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

where $x$ and $x'$ are variables and $a$ is a label. The semantics of $\mathrm{FT}_{\leq}$ is given by the interpretation over feature trees where the symbol $\leq$ is interpreted as information ordering on feature trees. Intuitively, the feature tree $\tau_1$ represents less information than $\tau_2$ if it has fewer edges and node labels. We formally define the relation $\leq$ in this sense. Algebraically, $\tau_1 \leq \tau_2$ holds if there is a *homomorphic embedding* from $\tau_1$ to $\tau_2$ (i.e., a mapping from nodes in $\tau_1$ to nodes in $\tau_2$ under which both the node labeling and the feature selection relation are invariant).

In contrast with the situation in previous feature constraint systems, the nodes of the feature trees in the interpretation domain of $\mathrm{FT}_{\leq}$ are possibly not labeled. This fact is significant when ordering constraints $x{\leq}y$ are involved (but is not, otherwise; the first-order theory of FT does not change whether we admit unlabeled nodes or not). The constraint $\exists z(z \leq x \wedge z \leq y)$ is valid in $\mathrm{FT}_{\leq}$. The information ordering $\leq$ has a smallest element, represented by the empty tree. The choice of this interpretation domain for $\mathrm{FT}_{\leq}$ has algorithmic consequences as well; i.e., the correctness of our algorithms depends on it. In an alternative semantics (see [25]), one may introduce a partial ordering on node labels; then, instead of having unlabeled nodes one could have a label that is smaller than all other labels.

It is clear that $\mathrm{FT}_{\leq}$ is as expressive as FT since the information ordering is antisymmetric (*i.e.*, $(x \leq x' \wedge x' \leq x) \leftrightarrow x = x'$ is valid). 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 algorithms in cubic time, one for the satisfiability problem ("$FT_{\leq} \models (\exists)\varphi$?") and one for the entailment problem ("$FT_{\leq} \models \varphi \rightarrow \varphi'$?") of the $\mathrm{FT}_{\leq}$ constraint system. We furthermore show that $\mathrm{FT}_{\leq}$ has the independence property. 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 $\mathrm{FT}_{\leq}$ constraints $\varphi \wedge \neg\varphi_1 \wedge \ldots \wedge \neg\varphi_n$ is decidable in $O(n^3)$. The test of entailment of constraints is a prerequisite for their use in modern constraint programming languages with advanced control mechanisms such as delaying, coroutining, synchronization, committed choice and local computation spaces [2, 5, 40, 32]. Thanks to the independence property, the entailment test together with the satisfiability test of positive constraints $\varphi$ 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 implications $\varphi \rightarrow \varphi_i$ is valid).

We reduce the satisfiability problem of Dörre's weak-subsumption constraints [14] over feature algebras linearly 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 [14].

**Plan of the Article.** Section 2 surveys related work. Section 3 defines $\mathrm{FT}_{\leq}$. Section 4 presents the satisfiability test for $\mathrm{FT}_{\leq}$ constraints. Section 5 presents the entailment test for $\mathrm{FT}_{\leq}$ constraints, and proves the independence property of $\mathrm{FT}_{\leq}$. Section 6 defines weak subsumption constraints and reduces their satisfiability problem to the one of $\mathrm{FT}_{\leq}$ constraints. Section 7 shows that $\mathrm{FT}_{\leq}$ is strictly more expressive than FT. Section 8 completes the proofs, and Section 9 concludes.

## 2. Related Work

**Ines Constraints.** In previous work [28], we have introduced the constraint system INES, whose constraints are inclusions between first-order terms interpreted over nonempty sets of trees; the satisfiability test is cubic. The satisfiability test for $\mathrm{FT}_{\leq}$ is inspired by the one for INES. The entailment problems for $\mathrm{FT}_{\leq}$ and INES constraints are different. Intuitively, the entailment problem of $\mathrm{FT}_{\leq}$ is less difficult than the one of INES because an $\mathrm{FT}_{\leq}$ constraint $\varphi$ cannot uniquely describe a single feature tree; in contrast, INES constraints 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 implication $x \subseteq a \rightarrow a \subseteq x$ holds in INES. The entailment problem of INES constraints is coNP-hard [26] (the algorithm given in [11] is not a complete test of entailment; the one given in [12] for a larger class of constraints lies in DEXPTIME).

**Feature Constraints.** The constraint system CFT [41] 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 [13] but provides finer-grained constraints. The system EF [42] extends CFT by feature constraints $x[y]z$, providing for first-class features. Complete axiomatizations for FT and CFT have been given in [8] and [7], respectively. The satisfiability of EF constraints is shown NP-hard in [42]. The system $\mathrm{FT}_\leq$(sort) extends $FT_\leq$ by allowing a partial order on labels [25]. In [27, 29] it is shown that the entailment problem for $\mathrm{FT}_\leq$ with existential quantifiers ("$FT_\leq \models \varphi \rightarrow \exists\bar{x}\varphi'$"), is PSPACE-complete, and that the first-order theory of $\mathrm{FT}_\leq$ is undecidable.

**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 [16, 14, 37]. 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 [16], Dörre proposed weak subsumption as a decidable approximation of subsumption. As we show in this article, 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 tree constraint systems RT, FT, CFT [13, 6, 4, 41], constraint systems with the independence property include linear equations over the real numbers [23], or infinite boolean algebras with positive constraints [17], and set constraints over nonempty sets of trees [28, 12].

## 3. Syntax and Semantics of $\mathrm{FT}_\leq$

The constraint system $\mathrm{FT}_\leq$ 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 $\varepsilon$ 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* $\tau$ is a pair $(D,L)$ consisting of a tree domain $D$ and partial labeling function $L : D \rightharpoonup \mathcal{L}$. Given a feature tree $\tau$, we write $D_\tau$ for its tree domain and $L_\tau$ 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* $\varphi$ is defined by the following abstract syntax.

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

An $\mathrm{FT}_\leq$ 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 $\mathrm{FT}_\leq$ constraints $\varphi$ up to associativity and commutativity of conjunction, *i.e.*, we view $\varphi$ as a multiset of inclusion, labeling, selection, and compatibility constraints. We write $\varphi$ *in* $\varphi'$ if all conjuncts in $\varphi$ are contained in $\varphi'$. The *size of a constraint* $\varphi$ is defined as the number of label and variable occurrences in $\varphi$.

**Semantics.** We next define the structure $\mathcal{F}$ over feature trees in which we interpret $\mathrm{FT}_\leq$ 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 $\Phi$ denote first-order formulae built from $\mathrm{FT}_\leq$ constraints with the usual first order connectives. We call $\Phi$ *satisfiable* (valid) if $\Phi$ is satisfiable (valid) in the structure $\mathcal{F}$. We say that $\Phi$ *entails* $\Phi'$, written $\Phi \models \Phi'$, if $\Phi \to \Phi'$ is valid, and that $\Phi$ is *equivalent* to $\Phi'$ if $\Phi_1 \leftrightarrow \Phi_2$ is valid. We denote with $V(\Phi)$ the set of variables occurring free in $\Phi$ and with $L(\Phi)$ the set of labels occurring in $\Phi$.

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

Proof: Let $\sigma$ 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 $\sigma$ is a solution of $x{\sim}y$. Conversely, if $\sigma$ 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 $\sigma'$ 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 $\mathrm{FT}_\leq$. The axioms and the algorithm are inspired by the ones for INES constraints presented in [28].

| | |
|---|---|
| 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 $\mathrm{FT}_{\leq}$ Constraints.

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 $\varphi$, an implication between constraints $\varphi \rightarrow \varphi'$, or an implication $\varphi \rightarrow \mathit{false}$.

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

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:

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

■

**The Algorithm F.** The set of axioms F induces a fixed point algorithm F that, given an input constraint $\varphi$, iteratively adds logical consequences of $\mathsf{F} \cup \{\varphi\}$ to $\varphi$. (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 $\varphi$ and terminates with *false* or outputs a constraint $\varphi \wedge \varphi'$. Termination with *false* takes place if there exists $\varphi''$ *in* $\varphi$ such that $\varphi'' \rightarrow \mathit{false} \in \mathsf{F}$. Output of $\varphi \wedge \varphi'$ is possible if $\varphi' \in \mathsf{F}$ or there exists $\varphi''$ *in* $\varphi$ with $\varphi'' \rightarrow \varphi' \in \mathsf{F}$.

EXAMPLE: 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: 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 $\mathsf{F}$ may add $x{\sim}y$ after several steps as shown in Example . Then it may proceed with $x{\sim}y'$ via $\mathsf{F4}$ and terminate with *false* via $\mathsf{F5}$. □

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

EXAMPLE: Our control takes care of termination in presence of cycles like $x[a]x$. For instance, the following constraint is $\mathsf{F}$-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 $\varphi$ is a constraint with $m$ variables then algorithm $\mathsf{F}$ with input $\varphi$ terminates under the above control in at most $2 \cdot m^2$ steps.*

Proof: Since $\mathsf{F}$ 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 $\mathsf{F}$-closed constraint $\varphi$ is satisfiable over $FT_{\leq}$.*

Proof: See Section 8.2. ■

THEOREM 1 *The satisfiability of $FT_{\leq}$ constraints can be decided in time $O(n^3)$ (offline and online, see [18]) where $n$ is the constraint size.*

Proof: Proposition 2 shows that $\varphi$ is unsatisfiable if $\mathsf{F}$ started with $\varphi$ terminates with *false*. Proposition 4 proves that $\varphi$ is satisfiable if $\mathsf{F}$ started with $\varphi$ terminates with a constraint. Since $\mathsf{F}$ terminates for all input constraints under the above control (Proposition 3) this yields a effective decision procedure. The main idea of the complexity proof is that one needs at most $O(n^2)$ steps (Proposition 3) each of which can be implemented in time $O(n)$. The implementation can be organized incrementally by exploiting that algorithm $\mathsf{F}$ 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 [28]. ■

## 5. Entailment, Independence, Negation

In this section, we give a cubic algorithm testing entailment $\varphi' \models \varphi$ between $\mathrm{FT}_{\leq}$ constraints $\varphi'$ and $\varphi$. We then prove the independence property of $\mathrm{FT}_{\leq}$. Hence we can solve conjunctions of positive and negative $\mathrm{FT}_{\leq}$-constraints $\varphi \wedge \neg\varphi_1 \wedge \ldots \neg\varphi_n$ in time $O(n^3)$.

A *basic constraint* $\mu$ is a conjunction free constraint $\varphi$, that is, 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 $\mu$ *in* $\varphi$.

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

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

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

PROPOSITION 5 (CORRECTNESS) *For all* $\mathsf{F}$*-closed constraints* $\varphi$ *and basic constraints* $\mu$*:* $\varphi \vdash \mu$ *implies* $\varphi \models \mu$.

Proof: Obvious. ■

We will show next that syntactic containment is strong enough to characterize entailment (Proposition 6) and that we can decide syntactic containment in time $O(n^3)$ for $\mathsf{F}$-closed constraints (Lemma 6). In combination with the cubic satisfiability test of the previous section, we obtain a cubic entailment test (Theorem 3).

We remains to show that syntactic containment is complete with respect to entailment: That is, that no primitive constraint $\mu$ is entailed by a constraint $\varphi$ if $\mu$ is not already contained in $\varphi$. To show this it suffices to find a solution of $\varphi$ that contradicts $\mu$. As we show, however, there exists a single solution that contradicts all such $\mu$ *at the same time*. We prove this by giving a satisfiable formula that strengthens $\varphi$ and entails the negation of all relevant $\mu$. Such a formula is called saturated and we construct one in Section 8.3.

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

1. $\mathrm{Sat}(\varphi)$ *is satisfiable.*
2. $\mathrm{Sat}(\varphi) \models \varphi$.
3. $\forall \mu$*: If* $V(\mu) \subseteq V(\varphi)$ *then* $\varphi \not\vdash \mu$ *implies* $\mathrm{Sat}(\varphi) \models \neg \mu$.

Proof: The proof is postponed to the end of Section 8.3. ■

THEOREM 2 (INDEPENDENCE) *The constraint system $FT_{\leq}$ has the following independence property: For every $n \geq 1$ and constraints $\varphi, \varphi_1, \ldots, \varphi_n$:*

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

Proof: Assume $\varphi \models \bigvee_{i=1}^{n} \varphi_i$. If $\varphi$ is unsatisfiable we are done. Also, if $\varphi \wedge \varphi_j$ is nonsatisfiable 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 $\varphi$ and $\varphi \wedge \varphi_i$ are satisfiable for all $i$, and that $\varphi$ is $\mathsf{F}$-closed. If there exists an $i$ such that $\varphi \vdash \mu$ for all $\mu \in \varphi_i$, then $\varphi \models \varphi_i$ by correctness of syntactic containment (Proposition 5) and we are done. Otherwise, for all $i$ there exists $\mu_i \in \varphi_i$ such that $\varphi \not\vdash \mu_i$. Let $\mathrm{Sat}(\varphi)$ be the formula postulated by Lemma 1. Since $V(\mu) \subseteq V(\varphi)$ for all $i$, we obtain by Property 3 that

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

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

PROPOSITION 6 (CHARACTERIZATION) *The notions of entailment and of syntactic containment coincide for basic constraints: If $\varphi$ is $\mathsf{F}$-closed and $\mu$ a basic constraint then $\varphi \models \mu$ iff $\varphi \vdash \mu$.*

Proof: Syntactic containment is *semantically correct* by Proposition 5. It remains to show that syntactic containment 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 $\mu$ 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 $\mathsf{F}$-closed constraint $\varphi$ of size $n$, we can compute a representation of $\varphi$ in time $O(n)$ that allows to test syntactic containment $\varphi \vdash \mu$ for selection constraints in time $O(n)$, and for all other $\mu$ in time $O(1)$.*

Proof: In [28] we have described in detail a data structure into which we can enter all primitive constraints of $\varphi$ in time $O(n)$ such that we can test in $O(1)$ whether a given labeling or inclusion constraint is syntactically contained in $\varphi$, and whether for a variable $x$ and a feature $f$ there exists a variable $y$ such that $x[f]y$ *in* $\varphi$. To check syntactic containment of $x[f]y$, $\varphi \vdash x[f]y$, we proceed as follows:

1. Check whether there exists $z \in V(\varphi)$ such that $\varphi \vdash z{\leq}x$, and than whether there exists $z'$ such that $z[f]z'$ *in* $\varphi$ and $y{\leq}z'$ *in* $\varphi$.

2. Inversely, check whether there exists $z \in V(\varphi)$ such that $\varphi \vdash x{\leq}z$, and then whether there exists $z'$ such that $z[f]z'$ *in* $\varphi$ and $z'{\leq}y$ *in* $\varphi$.

Clearly, $x[f]y$ is syntactically contained in $\varphi$ if and only if both checks succeed. Since there are two tests for every variable $z \in V(\varphi)$, this is a linear-time operation. ■

THEOREM 3 (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 $\varphi' \models \varphi$, we first test whether $\varphi'$ is satisfiable. By Theorem 1 this can be done by computing the $\mathsf{F}$-closure $\tilde{\varphi}'$ of $\varphi'$ in time $O(n^3)$. If this test fails then the entailment test is trivial. Otherwise, since $\tilde{\varphi}' \models \varphi$ if and only if $\varphi' \models \varphi$, we obtain from Proposition 6 that $\tilde{\varphi}' \models \varphi$ iff $\tilde{\varphi}' \vdash \mu$ for all $\mu$ *in* $\varphi$. Since there are $O(n)$ such $\mu$, this is decidable in time $O(n)$ by Lemma 2. The overall complexity sums up to $O(n^3)$. ■

COROLLARY 1 (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 $\varphi$ is non-satisfiable then $\varphi \wedge (\wedge_{i=1}^{n} \neg\varphi_i)$ is trivially non-satisfiable. By Proposition 1, satisfiability of $\varphi$ is decidable in time $O(n^3)$. Now assume $\varphi$ to be satisfiable. By the Independence Theorem 2, $\varphi \wedge (\wedge_{i=1}^{n} \neg\varphi_i)$ is nonsatisfiable if and only if $\varphi \models \varphi_i$ for some $i$. This is equivalent to saying that, for some $i$, $\mu$ *in* $\varphi_i$ implies $\varphi \models \mu$. By Proposition 6 it thus suffices to test syntactic containment in $\varphi$ for all $\mu$ contained in some $\varphi_i$. Overall, there are $O(n)$ such $\mu$'s to be tested for syntactic containment. By Lemma 2, $\varphi \vdash \mu$ 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 [14]. We show that their satisfiability problem is subsumed by the one for $\mathrm{FT}_{\leq}$.

**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 $\eta$ is a $\mathrm{FT}_{\leq}$ 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 [14]. 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 [39, 14] 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 7 *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 $\mathsf{F}$ and the antisymmetry of the information ordering in $\mathrm{FT}_{\leq}$ $(x{\leq}y \wedge y{\leq}x \rightarrow x{=}y)$. ■

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 $\Delta$ on the domain of $\mathcal{A}$ that satisfies the following properties for all elements $\alpha_1$, $\alpha_2$, $\alpha'_1$, $\alpha'_2$ of $\mathcal{A}$'s domain: check!

1. if $\alpha_1\Delta\alpha_2,\ c_1(\alpha_1)^{\mathcal{A}}$, then $c_1(\alpha_2)^{\mathcal{A}}$
2. if $\alpha_1\Delta\alpha_2,\ \alpha_1[d]^{\mathcal{A}}\alpha'_1$, then exists $\alpha'_2 : \alpha_2[d]^{\mathcal{A}}\alpha'_2$ 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 $\sim^{\mathcal{A}}$:

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

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 8 (DÖRRE [15]) *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 [15] on page 24 (Satz 6 and Satz 7). There the algebra 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 $\mathcal{F}$ due to the axioms in $\mathsf{F}$ and hence smaller than the weak subsumption relation of $\mathcal{F}$. Conversely, we show that every simulation on $\mathcal{F}$ is smaller then the information ordering. Let $\Delta$ be a simulation and let $\tau_1$ and $\tau_2$ be feature trees with $\tau_1\Delta\tau_2$. We must show that $\tau_1 \leq \tau_2$. This follows from $D_{\tau_1} \subseteq D_{\tau_2}$ and $L_{\tau_1} \subseteq L_{\tau_2}$ which can be proved by induction on paths. ■

THEOREM 4 *A weak subsumption constraint* $\eta$ *is satisfiable* $($*over* $\mathcal{F})$ *if and only if* $\eta$ *is satisfiable over the structure induced by some feature algebra* $\mathcal{A}$.

Proof: If $\eta$ 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 $\mathsf{F}$. Thus, if $\eta$ is satisfiable over one such structure then it is equivalent to an $\mathsf{F}$-closed constraint (and not *false*) and hence satisfiable over $\mathcal{F}$. ■

**Alternative Notions of Feature Algebras.** In the literature [39, 14] 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 [14].

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 9 *A constraint* $\eta$ *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 $\eta$ is satisfiable over the feature algebra $\mathcal{F}$ with labels $\mathcal{C}\cup\mathcal{D}$. Given a solution $\sigma'$ of $[\![\eta]\!]$ over $\mathcal{A}$ a solution $\sigma$ of $\eta$ over $\mathcal{F}$ can be defined as follows:

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

Conversely, let $\eta$ be satisfiable in a feature algebra $\mathcal{A}$. Then $\eta$ is satisfiable in $\mathcal{F}$ by Theorem 4. 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 $\tau$ without labeled internal nodes where a *labeled internal node* of $\tau$ 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 $\sigma$ be an $\mathcal{A}$-solution of $\eta$. Then the variable assignment $\sigma'$ mapping $x$ on $\sigma'(x)$ as given below is an $\mathcal{F}^*$-solution of $[\![\eta]\!]$.

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

■

## 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 $\eta$ is of the form $x{=}y$, $a(x)$, $x[a]y$, or $\eta \wedge \eta'$. 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 10 *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 $\varphi$ were such a $\mathrm{FT}_{\leq}$ constraint, then $\varphi$ as well as its finite $\mathsf{F}$-closure would entail $x{\leq}y$ for all variables $y$. This contradicts Proposition 6 for all those $y$ such that $y \notin V(\varphi)$ and $x \neq y$ (because if $\varphi \vdash x{\leq}y$ then $x = y$ or $x, y \in V(\varphi)$). Such a variable $y$ exists since $V(\varphi)$ is finite. ■

LEMMA 3 *Let $\eta$ be an $FT$ constraint. Then $\eta \models x{\leq}y$ if and only if $\eta \models y{\leq}x$.*

Proof: The $FT$ constraint $\eta$ is equivalent to the $FT_{\leq}$ constraint $\varphi$ obtained from $\eta$ by replacing all equalities $x{=}y$ by inequalities $x{\leq}y \wedge y{\leq}x$. Hence, $x{\leq}y$ *in* $\varphi$ iff $y{\leq}x$ *in* $\varphi$, and since algorithm $\mathsf{F}$ preserves this invariant it also holds for the $\mathsf{F}$-closure of $\varphi$. The claim follows from Proposition 6. ■

PROPOSITION 11 *If $x \neq y$ then there is no $FT$ constraint $\eta$ equivalent to $x{\leq}y$.*

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

## 8. Proofs

We complete the completeness of the satisfiability check and the existence of a saturated formula as postulated in Lemma 1.

### 8.1. Path Reachability

In both proofs we need the following notion. For all paths $p$ and constraint $\varphi$, we define a binary relation $\leadsto_p^{\varphi}$, where $x \leadsto_p^{\varphi} y$ reads as "$y$ is reachable from $x$ over path $p$ in $\varphi$":

$$\begin{array}{lll} x \leadsto_{\varepsilon}^{\varphi} y & \text{if} & y{\leq}x \text{ in } \varphi \\ x \leadsto_{a}^{\varphi} y & \text{if} & x[a]y \text{ in } \varphi, \\ x \leadsto_{pq}^{\varphi} y & \text{if} & x \leadsto_{p}^{\varphi} z \text{ and } z \leadsto_{q}^{\varphi} y. \end{array}$$

Define relationships $x \leadsto_p^{\varphi} a$ meaning that "$a$ can be reached from $x$ over path $p$ in $\varphi$":

$$x \rightsquigarrow^{\varphi}_{p} a \quad \text{if} \quad x \rightsquigarrow^{\varphi}_{p} y \text{ and } a(y) \textit{ in } \varphi,$$

For example, if $\varphi$ 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: $y \rightsquigarrow^{\varphi}_{\varepsilon} x$, $x \rightsquigarrow^{\varphi}_{b} z$, $x \rightsquigarrow^{\varphi}_{ba} y$, $x \rightsquigarrow^{\varphi}_{ba} x$, *etc.*, as well as $x \rightsquigarrow^{\varphi}_{\varepsilon} a$, $x \rightsquigarrow^{\varphi}_{b} b$, $x \rightsquigarrow^{\varphi}_{ba} a$, *etc.*

### 8.2. Completeness of the Satisfiability Test

We show the following result.

PROPOSITION 4 *Every* $\mathsf{F}$*-closed constraint* $\varphi$ *is satisfiable over* $FT_{\leq}$.

Proof: If $\varphi$ is $\mathsf{F}$-closed then $\varphi$ is path consistent by Lemma 5 and thus satisfiable by Lemma 4. ■

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

1. *If* $x \rightsquigarrow^{\varphi}_{p} a$, $x \leq x$, *and* $x \rightsquigarrow^{\varphi}_{p} b$ *then* $a = b$.
2. *If* $x \rightsquigarrow^{\varphi}_{p} a$, $x \sim y$, *and* $y \rightsquigarrow^{\varphi}_{p} b$ *then* $a = b$.

LEMMA 4 *Every* $\mathsf{F1}$-$\mathsf{F2}$*-closed and path consistent constraint is satisfiable.*

Proof: Let $\varphi$ be $\mathsf{F1}$-$\mathsf{F2}$-closed and path consistent. We define the variable assignment $\mathsf{min}_{\varphi}$ into feature trees as follows:

$$D_{\mathsf{min}_{\varphi}(x)} = \{p \mid x \rightsquigarrow^{\varphi}_{p} y\} \qquad \text{and} \qquad L_{\mathsf{min}_{\varphi}(x)} = \{(p,a) \mid x \rightsquigarrow^{\varphi}_{p} a\} \tag{2}$$

The path consistency of $\varphi$ 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 $\mathsf{min}_{\varphi}$ is a solution of $\varphi$.

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

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

  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 $y \rightsquigarrow^{\varphi}_{p} z$ then $x \rightsquigarrow^{\varphi}_{ap} z$ since $x[a]y$ *in* $\varphi$. Suppose $x \rightsquigarrow^{\varphi}_{ap} z$. By definition of path reachability there exists $x'$ and $y'$ such that

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

The F1-closedness of φ and $x \rightsquigarrow^{\varphi}_{\varepsilon} x'$ imply $x \leq x'$ *in* φ. The F2-closedness ensures $y \leq y'$ *in* φ such that $y \rightsquigarrow^{\varphi}_{p} z$ holds. We now prove the second equivalence above. If $x \rightsquigarrow^{\varphi}_{ap} b$ then there exists $z$ such that $x \rightsquigarrow^{\varphi}_{ap} z$ and $b(z)$. The first equivalence implies $y \rightsquigarrow^{\varphi}_{p} z$ and thus $y \rightsquigarrow^{\varphi}_{p} b$. The converse is simple.

- Let $a(x)$ in φ. Reflexivity (F1.1-closedness) implies $x \leq x$ *in* φ. Thus $x \rightsquigarrow^{\varphi}_{\varepsilon} a$ 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 $x \rightsquigarrow^{\varphi}_{p} a$ and $y \rightsquigarrow^{\varphi}_{p} b$. The path consistency of φ condition 2 implies $a = b$. ■

LEMMA 5 *Every* F3, F4, F5*-closed constraint is path consistent.*

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

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

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 $p = a'q$, then there exists there exist $x'$, $y'$, $\tilde{x}$, $\tilde{y}$ with:

$$x \rightsquigarrow^{\varphi}_{\varepsilon} x', \quad x'[a']\tilde{x} \; in \; \varphi, \quad \tilde{x} \rightsquigarrow^{\varphi}_{p} a, \quad \text{and} \quad y \rightsquigarrow^{\varphi}_{\varepsilon} y', \quad y'[a']\tilde{y} \; in \; \varphi, \quad \tilde{y} \rightsquigarrow^{\varphi}_{p} b .$$

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. ■

COROLLARY 2 *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 $\mathsf{min}_\varphi$ of an F-closed constraint that $a \notin L_{\mathsf{min}_\varphi(y)}$. Hence $\varphi \not\models \exists y(x[a]y)$. ■

## 8.3. Saturation

We prove the existence of a saturated formula Sat(φ) as postulated in Lemma 1.

We construct Sat(φ) by means of two operators $\Gamma_1$ and $\Gamma_2$ on constraints. The operator $\Gamma_2$ is such that $\Gamma_2(\varphi)$ disentails all $\mu$ 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 8). The operator $\Gamma_1$ is necessary to also disentail selection constraints. Given a constraint φ, $\Gamma_1(\varphi)$ extends φ such that $\Gamma_2$ applied to the $\mathsf{F}$-closure suffices to disentail all relevant $\mu$. In a sense, $\Gamma_1$ is a "preprocessor" for $\Gamma_2$.

DEFINITION ($\Gamma_1$ AND $\Gamma_2$) *Let $\varphi$ be a constraint, $v_1$ and $v_2$ distinct fresh variables, and $a_1$ and $a_2$ distinct labels. Furthermore, for every pair of variables $x, y \in V(\varphi)$, and every label $a \in L(\varphi)$ let $a_x$ and $a_{xy}$ be fresh labels and $v_{xa}$ a fresh variable. We define $\Gamma_1(\varphi)$ and $\Gamma_2(\varphi)$ depending on $v_1, v_2, a_1, a_2, a_x, a_{xy}, v_{xa}$ as follows:*

$$\begin{array}{rcll}
\Gamma_1(\varphi) & = & \varphi \wedge \bigwedge\{x[a]v_{xa} \mid x \in V(\varphi),\, a \in L(\varphi)\} & \\
\Gamma_2(\varphi) & = & \varphi \;\wedge\; \bigwedge\{x[a_x]v_x \wedge \neg\exists y'(y[a_x]y') \mid \varphi \not\models x \leq y,\; x, y \in V(\varphi)\} & (1)\\
& & \wedge\; \bigwedge\{x[a_{xy}]v_1 \wedge y[a_{xy}]v_2 \mid \varphi \not\models x \sim y,\; x, y \in V(\varphi)\} & (2)\\
& & \wedge\; \bigwedge\{x \sim v_1 \wedge x \sim v_2 \mid \textit{for all } a \in L:\ \varphi \not\models a(x),\; x \in V(\varphi)\} & (3)\\
& & \wedge\; a_1(v_1) \wedge a_2(v_2) & (4)
\end{array}$$

EXAMPLE: For illustration of $\Gamma_1$ and $\Gamma_2$ consider the constraint $\varphi_{contra} =_{def} x[a]x \wedge y \leq x$, which is $\mathsf{F}$-closed up to trivial constraints and which does not entail $x[a]y$. In order to disentail $x[a]y$ we first add $x[a]v_{xa}$ and $y[a]v_{ya}$ to $\varphi_{contra}$ according to $\Gamma_1$. The $\mathsf{F}$-closure of $\Gamma_1(\varphi_{contra})$ is (again up to trivial constraints)

$$\Gamma'_1(\varphi_{contra}) = x[a]x \wedge y \leq x \wedge x[a]v_{xa} \wedge y[a]v_{ya} \wedge v_{ya} \leq v_{xa} \wedge v_{xa} \leq x \wedge x \leq v_{xa} \wedge y \leq v_{xa}$$

Observe that $\Gamma'_1(\varphi_{contra})$ does not contain $v_{xa} \leq y$; that is, $\Gamma'_1(\varphi_{contra}) \not\models v_{xa} \leq y$. Now clause (1) of $\Gamma_2(\Gamma'_1(\varphi_{contra}))$ disentails $v_{xa} \leq y$ by asserting that $y$ allows selection at feature $a_y$ while $v_{xa}$ does not. Hence, $\Gamma_2(\Gamma'_1(\varphi_{contra}))$ also disentails $x[a]y$. □

LEMMA 6 (PROPERTIES OF $\Gamma_1$) *Let $\varphi$ be an $\mathsf{F}$-closed (and hence satisfiable) constraint. Then $\Gamma_1(\varphi)$ is satisfiable and its $\mathsf{F}$-closure $\Gamma'_1(\varphi)$ satisfies the following two properties for all primitive constraints $\mu$:*

1. *If $\varphi \not\models \mu$ and $V(\mu) \subseteq V(\varphi)$, then $\Gamma'_1(\varphi) \not\models \mu$.*
2. *If $\varphi \not\models x[a]y$, then $\Gamma'_1(\varphi) \not\models y \leq v_{xa}$ or $\Gamma'_1(\varphi) \not\models v_{xa} \leq y$.*

Proof: The $\mathsf{F}$-closure $\Gamma'_1(\varphi)$ of $\Gamma_1(\varphi)$ has the following form up to trivial constraints and symmetry of compatibility constraints.

$$\Gamma'_1(\varphi) = \Gamma_1(\varphi) \wedge \bigwedge\{v_{xa} \leq v_{ya} \mid \varphi \vdash x \leq y, a \in L(\varphi)\} \tag{4.1}$$

$$\wedge \bigwedge\{v_{xa} \sim v_{ya} \mid \varphi \vdash x \sim y, a \in L(\varphi)\} \tag{4.2}$$

$$\wedge \bigwedge\{z \leq v_{xa} \mid \text{exists } y, y' : z \leq y' \wedge y[a]y' \wedge y \leq x \in \varphi\} \tag{4.3}$$

$$\wedge \bigwedge\{v_{xa} \leq z \mid \text{exist } y, y' : x \leq y \wedge y[a]y' \wedge y' \leq z \in \varphi\} \tag{4.4}$$

$$\wedge \bigwedge\{v_{xa} \sim z \mid \text{exist } y, y' : x \leq y \wedge y[a]y' \wedge y' \sim z \in \varphi\} \tag{4.5}$$

$$\wedge \bigwedge\{v_{xa} \sim z \mid \text{exist } y, y' : x \sim y \wedge y[a]y' \wedge z \leq y' \in \varphi\} \tag{4.6}$$

It is clear that $\Gamma'_1(\varphi)$ is contained in the F-closure of $\Gamma_1(\varphi)$. For instance in the case of Clause (4.3), if $z \leq y' \wedge y[a]y' \wedge y \leq x \in \varphi$, then $z \leq v_{xa}$ can be derived from $\varphi \wedge x[a]v_{xa}$ using axioms F2 and F1.2, and hence from $\Gamma_1(\varphi)$.

The F-closedness of $\Gamma'_1(\varphi)$ is proved by a tedious case distinction. F1.1 and F3.3 are clear by assumption, and it is simple to check that none of the clash axioms applies to $\Gamma'_1(\varphi)$, essentially because $\Gamma'_1(\varphi)$ contains no labeling constraints on fresh variables. We check the remaining cases.

F1.2 Assume $x \leq y \wedge y \leq z \in \Gamma'_1(\varphi)$. We must show that $x \leq z \in \Gamma'_1(\varphi)$. We make a case distinction depending on which of the variables $x, y, z$ already occur in $V(\varphi)$, and which ones are of the form $v_{ua}$ for some $u$ and $a$.

If $x, y, z \in V(\varphi)$ then $x \leq y \wedge y \leq z \in \varphi$. This implies $x \leq z \in \varphi$ since $\varphi$ is F1.2-closed, and hence $x \leq z \in \Gamma'_1(\varphi)$.

If $x, y, z \notin V(\varphi)$ then the inclusions $x \leq y$ and $y \leq z$ have been added by Clause (4.1). Thus there exist variables $x', y', z'$ and a sort $a$ such that $x = v_{x'a}$, $y = v_{y'a}$, $z = v_{z'a}$, and $x' \leq y' \wedge y' \leq z' \in \varphi$. Since $\varphi$ is F1.2-closed, $x' \leq z' \in \varphi$, and so, by Clause (4.1) again, $x \leq z \in \Gamma'_1(\varphi)$.

If $x = v_{ua}$ and $y, z \in V(\varphi)$ then the inclusion $v_{ua} \leq y$ must have been added by Clause (4.4). This implies that there exist variables $v, v'$ such that $x \leq v \wedge v[a]v' \wedge v' \leq y \in \varphi$. Since $\varphi$ is F-closed this implies that $v' \leq z \in \varphi$ from F1.2 and $y \leq z \in \varphi$, and hence $x = v_{ua} \leq z \in \Gamma'_1(\varphi)$ by Clause (4.4) again.

If $z = v_{ua}$ and $x, y \in V(\varphi)$ then an argument similar to the previous case applies, based on Clause (4.3) instead of Clause (4.4).

If $y = v_{ua}$ and $x, z \in V(\varphi)$ then there exist variables $v, v'$ and $w, w'$ such that $u \leq v \wedge v[a]v' \wedge v' \leq z \in \varphi$, as well as $x \leq w' \wedge w[a]w' \wedge w \leq u \in \varphi$ by Clauses (4.3) and (4.4). By F-closedness of $\varphi$ we obtain

$$\begin{array}{ll} w \leq v \in \varphi & \text{from } w \leq u, u \leq v, \text{ and F1.2} \\ w' \leq v' \in \varphi & \text{from } w[a]w', v[a]v', \text{ and F2, and then} \\ x \leq z \in \varphi & \text{from } x \leq w', v' \leq z, \text{ and F1.2.} \end{array}$$

... The remaining cases that exactly one of $x,y,z$ is in $V(\varphi)$ are similar.

F3.2 Similar to the previous case.

F3.1 Based on F-closedness of $\varphi$ it is easy to check that the condition in Clauses (4.5) and (4.6) hold if the conditions in Clauses (4.3) and (4.4) hold, respectively. This proves F3.1-closedness up to (the assumed) symmetry of non-disjointness.

F2 Assume $x[a]x' \wedge x{\leq}y \wedge y[a]y' \in \Gamma_1'(\varphi)$. We must show that $x'{\leq}y' \in \Gamma_1'(\varphi)$. Since $\Gamma_1'(\varphi)$ does not contain any selection constraint for a fresh variable, we know that $x,y \in V(\varphi)$.

- If both $x',y'$ are fresh variables (*i.e.*, $x' = v_{xa}$ and $y' = v_{ya}$) then $x'{\leq}y' \in \Gamma_1'(\varphi)$ due to Clause (4.1).
- If $x' = v_{xa}$ and $y' \in V(\varphi)$, then $x' = v_{xa}{\leq}y'$ follows from Clause (4.4) using that $\varphi \vdash y'{\leq}y'$.
- Dual to the previous case using Clause (4.3) instead of Clause (4.4).

F5 Similar to the previous case, using Clause (4.2) instead of (4.1).

Now we check properties (1) and (2) of $\Gamma_1'(\varphi)$. In both cases, we prove the contraposed claim.

1. Assume that $\Gamma_1'(\varphi) \vdash \mu$ and $V(\mu) \subseteq V(\varphi)$. We show that $\varphi \vdash \mu$ by case distinction over $\mu$.

   **$\mu = x{\leq}y$ or $\mu = x{\sim}y$:** If $\Gamma_1'(\varphi) \vdash \mu$ then $\mu \in \Gamma_1'(\varphi)$ or $x = y$. If $x = y$, then trivially $\varphi \vdash \mu$. Otherwise, if $x \neq y$, note that all primitive constraints which are contained in $\Gamma_1'(\varphi)$ but not in $\varphi$ contain at least one fresh variable. Hence from $x,y \in V(\mu) \subseteq V(\varphi)$ we obtain $\mu \in \varphi$, and therefore $\varphi \vdash \mu$.

   **$\mu = a(x)$:** If $\Gamma_1'(\varphi) \vdash a(x)$ then there exists a variable $x'$ such that $x{\leq}x' \wedge a(x') \in \Gamma_1'(\varphi)$. By inspection of the form of $\Gamma_1'(\varphi)$ one obtains that $x' \in V(\varphi)$. In combination with the assumption that $V(\mu) \subseteq V(\varphi)$ this implies $x \in V(\varphi)$ and therefore $\varphi \vdash a(x)$.

   **$\mu = x[a]y$:** If $\Gamma_1'(\varphi) \vdash x[a]y$ then there exist variables $u,u'$ and $v,v'$ such that

   $$\Gamma_1'(\varphi) \vdash u{\leq}x \wedge x{\leq}v\,,$$
   $$\Gamma_1'(\varphi) \vdash y{\leq}u' \wedge v'{\leq}y\,, \text{ and}$$
   $$u[a]u' \wedge v[a]v' \in \Gamma_1'(\varphi)$$

   By assumption, $x,y \in V(\mu) \subseteq V(\varphi)$. Also $u,v \in V(\varphi)$ holds since $\Gamma_1'(\varphi)$ contains no selection constraints on fresh variables.
   We can without loss of generality assume that $u',v' \in V(\varphi)$. In this case $\varphi \vdash x[a]y$ follows easily.
   To see why we can assume $u',v' \in V(\varphi)$, suppose $u' \notin V(\varphi)$. Then $u' = v_{uf}$ by construction of $\Gamma_1$, and by Clause (4.3) there must exist variables $w,w' \in V(\varphi)$ such that $y{\leq}w' \wedge w[a]w' \wedge w{\leq}x \in \varphi$. Hence, we can replace $w,w'$ for $u,u'$ above and obtain the same situation up to renaming. The argument for $v'$ is dual.

2. Assume that $\Gamma'_1(\varphi) \vdash z \leq v_{xa}$ and $\Gamma'_1(\varphi) \vdash v_{xa} \leq z$. Then by Clauses (4.3) and (4.4) there must exist variables $y, y'$ and $u, u'$ such that

$$z \leq y' \wedge y[a]y' \wedge y \leq x \in \varphi, \quad \text{and} \quad x \leq u \wedge u[a]u' \wedge u' \leq z \in \varphi\,.$$

By definition of syntactic containment these assumptions imply $\varphi \vdash x[a]z$. ■

LEMMA 7 ($\Gamma_2$ PRESERVES SATISFIABILITY) *If $\varphi$ is $\mathsf{F}$-closed, then $\Gamma_2(\varphi)$ is satisfiable.*

Proof: Let $\varphi_\Gamma$ be the constraint part of $\Gamma_2(\varphi)$ (*i.e.*, with existential quantifiers and negated selection constraints dropped). It is not difficult to show that $\varphi_\Gamma$ is $\mathsf{F}$-closed up to trivial constraints ($x \leq x$ and $x \sim x$) and symmetric compatibility constraints. Note in particular, that the fresh features $a_x$ occur only once in $\Gamma_2(\varphi)$ (and hence neither $\mathsf{F2}$ nor $\mathsf{F5}$ applies), and that the fresh features $a_{xy}$ occur exactly twice in $\Gamma_2(\varphi)$, namely in selections at $x$ and $y$, for which neither $x \sim y$ nor, by $\mathsf{F3.1}$-closedness of $\varphi$, $x \leq y$ or $y \leq x$ occur in $\varphi$.

Hence $\mathsf{min}_{\varphi_\Gamma}$ as defined the proof of Lemma 4 in Section 8.2 is a solution of $\varphi_\Gamma$. It suffices to check that $\mathsf{min}_{\varphi_\Gamma}$ also satisfies the negated selection constraints added in clause (1) of $\Gamma_2(\varphi)$.

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

LEMMA 8 ($\Gamma_2$ CONTRADICTS NON-SELECTION CONSTRAINTS) *Let $\varphi$ be an $\mathsf{F}$-closed constraint and let $\mu$ be a primitive constraint of the form $x \sim y$, $x \leq y$, or $a(x)$. 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 7 and correctness of syntactic containment. 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 $a_x$ 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 $a_{xy}$ such that the subtrees of $x$ and $y$ at $a_{xy}$ 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 contain at least two trees with distinct label. ■

DEFINITION (SATURATION) *Let $\varphi$ be an $\mathsf{F}$-closed constraint and let $\Gamma'_1(\varphi)$ be the $\mathsf{F}$-closure of $\Gamma_1(\varphi)$. By Lemma 6, $\Gamma'_1(\varphi)$ is satisfiable so that its $\mathsf{F}$-closure exists. Then we define a concrete saturation* $\mathrm{Sat}(\varphi)$ *of $\varphi$ by* $\mathrm{Sat}(\varphi) =_{def} \Gamma_2(\Gamma'_1(\varphi))$.

LEMMA 9 (SATURATION CHARACTERIZES SYNTACTIC ENTAILMENT) *Let $\varphi$ be an $\mathsf{F}$-closed constraint and let $\mu$ be such that $V(\mu) \subseteq V(\varphi)$. Then $\varphi \not\vdash \mu$ implies* $\mathrm{Sat}(\varphi) \models \neg\mu$.

Proof: Let $\Gamma'_1(\varphi)$ the $\mathsf{F}$-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 case (1) of Lemma 6. If $\mu$ is not a selection constraint, then $\Gamma_2(\Gamma'_1(\varphi)) \models \neg\mu$

by Lemma 8. Otherwise, let $\mu = x[a]y$. Hence, $\Gamma'_1(\varphi) \not\models v_{xa} \leq y$ or $\Gamma'_1(\varphi) \not\models y \leq v_{xa}$ by case (2) of Lemma 6. By Lemma 8, 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, and hence again $\Gamma_2(\Gamma'_1(\varphi)) \models \neg\mu$. ∎

Proof of Lemma 1: We check that Sat($\varphi$) has the three postulated properties. (1) The saturation formula Sat($\varphi$) entails $\varphi$ by construction. (2) Lemmas 6 and 7 prove that Sat($\varphi$) is satisfiable. (3) By Lemma 9, Sat($\varphi$) contradicts all $\mu$ that $\varphi$ does not contain syntactically.

## 9. Conclusion

We have presented and investigated the constraint system $FT_{\leq}$ which adds information ordering constraints to the constraint system FT over feature trees. We have considered satisfiability and entailment problems for $FT_{\leq}$ constraints. We have shown that satisfiability problem of positive and negative $FT_{\leq}$ constraints with existential quantification is NP-complete, but that two important subproblems are decidable in $O(n^3)$ and we have given algorithms for both. Furthermore, we have shown that our satisfiability test for positive $FT_{\leq}$ constraints improves the known complexity of the satisfiability problem for weak subsumption constraints.

## 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 Esprit Working Group CCL II (EP 22457) and the Deutsche Forschungsgemeinschaft DFG through the SFB 378 at the Universität des Saarlandes.

## 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*, 3:185–215, 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 and 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. A complete axiomatization of a theory with feature and arity constraints. *The Journal of Logic Programming*, 24(1 and 2):37–71, 1995. Special Issue on Computational Linguistics and Logic Programming.

8. Rolf Backofen and Gert Smolka. A complete and recursive feature theory. *Theoretical Computer Science*, 146(1–2):243–268, July 1995.
9. 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.
10. 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.
11. 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*. Springer-Verlag, Berlin, 1996.
12. 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.
13. 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.
14. Jochen Dörre. Feature-logic with weak subsumption constraints. In *Constraints, Languages, and Computation*, chapter 7, pages 187–203. Academic Press, 1994.
15. Jochen Dörre. *Feature-Logik und Semiunifikation*. Number 128 in DISKI - Dissertationen zur Künstlichen Intelligenz. Infix Verlag, Sankt Augustin, July 1996. In German.
16. Jochen Dörre and William C. Rounds. On subsumption and semiunification in feature algebras. In *Proceedings of the 5$^{th}$ IEEE Symposium on Logic in Computer Science*, pages 300–310. IEEE Computer Society Press, 1990.
17. 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.
18. Joxan Jaffar and Michael J. Maher. Constraint logic programming: A survey. *The Journal of Logic Programming*, 19/20:503–582, May-July 1994.
19. Mark Johnson. *Attribute-Value Logic and the Theory of Grammar*. Number 16 in CSLI Lecture Notes. CSLI, 1988.
20. 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.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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, to appear. Springer-Verlag, Berlin, 1997.
26. 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`.
27. 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*, Lecture Notes in Computer Science, Tsukuba, Japan, 1998. Springer-Verlag, Berlin.
28. Martin Müller, Joachim Niehren, and Andreas Podelski. Inclusion constraints over non-empty sets of trees. In Michel Bidoit and Max Dauchet, editors, *Theory and Practice of Software Development*, volume 1214 of *Lecture Notes in Computer Science*, pages 345–356, Lille, France, April 1997. Springer-Verlag, Berlin.
29. Martin Müller, Joachim Niehren, and Ralf Treinen. The first-order theory of ordering constraints over feature trees. In Vaughan R. Pratt, editor, *Proceedings of the 13$^{th}$ IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1998.

30. Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, 1990.
31. 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.
32. Andreas Podelski and Gert Smolka. Operational semantics of constraint logic programs with coroutining. In Leon Sterling, editor, *Proceedings of the International Conference on Logic Programming*, pages 449–463, Kanagawa, Japan, 13–18 June 1995. The MIT Press, Cambridge, MA.
33. Carl Pollard and Ivan Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. Cambridge University Press, Cambridge, England, 1994.
34. Carl J. Pollard and Ivan A. Sag. *Information-based Syntax and Semantics, Vol. 1*. Number 13 in CSLI Lecture Notes. CSLI Publications, Stanford University, 1987. Distributed by University of Chicago Press.
35. William C. Rounds. Feature logics. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*. Elsevier Science Publishers B.V. (North Holland), 1997.
36. Steward Shieber. *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes No. 4. Center for the Study of Language and Information, 1986.
37. Steward Shieber. *Parsing and Type Inference for Natural and Computer Languages*. SRI International Technical Note 460, Stanford University, March 1989.
38. 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.
39. Gert Smolka. Feature constraint logics for unification grammars. *The Journal of Logic Programming*, 12:51–87, 1992.
40. 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.
41. Gert Smolka and Ralf Treinen. Records for logic programming. *The Journal of Logic Programming*, 18(3):229–258, April 1994.
42. 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.
43. Andreas Zeller and Gregor Snelting. Handling version sets through feature logic. In W. Schäfer and P. Botella, editors, *Proceedings of the $5^{th}$ European Software Engineering Conference*, volume 989 of *Lecture Notes in Computer Science*, pages 191–204, Sitges, Spain, September 1995. Springer-Verlag, Berlin.
44. Andreas Zeller and Gregor Snelting. Unified versioning through feature logic. *ACM Transactions on Software Engineering and Methodology*, 6(4):398–441, October 1997.