A Typed Feature Logic with Set-valued Attributes as a Foundation for LP Rules

Natali I. Alt

Geissweg 19
72076 Tübingen

Schriftliche Hausarbeit zur Erlangung des akademischen Grades eines Magister Artium

Gutachter: Prof. E. W. Hinrichs und Prof. U. Mönnich

21.Oktober 1996

Hiermit versichere ich, die Magisterarbeit selbständig verfaßt zu haben. Zur Erstellung der Arbeit habe ich keine anderen Hilfsmittel als die angegebenen Quellen benutzt.

Tübingen, 21.10.1996 Natali I. Alt

Contents

1 Introduction

Although Pollard and Sag (1987) and Pollard and Sag (1994) introduce *linear precedence* (LP) constraints for *Head-Driven Phrase Structure Grammar* (HPSG), it is not clear how they are to be integrated in an underlying formalism, a feature logic. One possible approach[1] to model LP constraints is presented in (Manandhar 1995b) and (Manandhar 1995a): The feature logic is enriched by relational (or set-valued) attributes with the aim of modeling precedence relations and set descriptions, respectively, by means of feature terms with special relational features.

Both (Manandhar 1995a) and (Manandhar 1995b) are concerned with an untyped feature logic and provide a consistency checking procedure for the respective logic. In this thesis, I will show how set-valued attributes can be added to a *typed* feature logic.

In the subsequent section, I will introduce a typed feature logic on which the relational extensions will be based. How set-valued attributes can be integrated in the logic is the topic of the third section. The fourth part constitutes the main content of the thesis. It supplies a decision procedure to check the satisfiability of feature terms of the extended logic. The last section draws the connection to LP rules and outlines how the description language could be used to model word order constraints and which further extensions would be desirable.

2 A Typed Feature Logic

The feature logic described in this and extended in the next section is mainly based on (Smolka 1988) and on the *Speciate Re-entrant Logic* (SRL) of King (1989) and provides a mathematical formalism for HPSG (*Head-Driven Phrase Structure Grammar*), which was developed in (Pollard and Sag 1987) and (Pollard and Sag 1994). A feature logic like SRL offers the possibility to write down descriptions in a formal language. Those descriptions denote sets of objects of a universe, so a description can be either true or false of an object.

Definition 2.1 (Signature) *A signature is a quadruple* $\langle \mathcal{T}, \preceq, \mathcal{F}, approp \rangle$ *s.t.*

- $\mathcal{T}$ *is a finite set of types, and* $\langle \mathcal{T}, \preceq \rangle$ *is a join semi-lattice*
- $\mathcal{V} = \{t \in \mathcal{T} \mid \textit{if } t' \preceq t \textit{ then } t' = t\}$ *is the set of minimal types.*

[1] Richter and Sailer (1995) also provide a possibility to express LP rules in HPSG. They assume an acyclic `PHON` list and show that LP rules can be expressed by recursive sorts in SRL (King (1989)).

- $\mathcal{F}$ *is a finite set of feature names*

- $\text{approp} : \mathcal{V} \times \mathcal{F} \rightharpoonup \mathcal{T}$ *is a partial function from pairs of minimal types and features to types*

Also, VAR *is a countably infinite set of variables.*

The minimal types will also be called *varieties* or *species*, *glb* will be shorthand for the *greatest lower bound* in $\mathcal{T}$, and we will use the symbol $\top$ for the most general type: For each $t \in \mathcal{T}$, $t \preceq \top$.

Definition 2.2 (Feature Terms) *The set of (feature) terms is the smallest set s.t.*

- x *is a term* *if* $x \in \mathsf{VAR}$.

- t *is a term* *if* $t \in \mathcal{T}$.

- $f : \phi$ *is a term* *if* $f \in \mathcal{F}$ *and* ϕ *is a term.*

- $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg\phi_1$ *are terms* *if* ϕ_1 *and* ϕ_2 *are terms.*

Definition 2.3 (Interpretation) *An interpretation is a triple* $\mathcal{I} = \langle \mathcal{U}, \mathcal{S}, \mathcal{A} \rangle$ *s.t.*

- $\mathcal{U}$ *is a set of objects, the domain of* $\mathcal{I}$

- $\mathcal{S} : \mathcal{U} \to \mathcal{V}$ *is a total function from the set of objects to the set of minimal types*

- $\mathcal{A} : \mathcal{F} \to \mathcal{U}^{\mathcal{U}}$ *is an attribute interpretation function s.t.*

  - *for each* $u \in \mathcal{U}$*, for each* $f \in \mathcal{F}$*, if* $\text{approp}(\mathcal{S}(u), f)$ *is defined*[2] *and* $\text{approp}(\mathcal{S}(u), f) = t$*, then* $\mathcal{A}(f)(u)$ *is defined and* $\mathcal{S}(\mathcal{A}(f)(u)) \preceq t$

  - *for each* $u \in \mathcal{U}$*, for each* $f \in \mathcal{F}$*, if* $\mathcal{A}(f)(u)$ *is defined, then* $\text{approp}(\mathcal{S}(u), f)$ *is defined and* $\mathcal{S}(\mathcal{A}(f)(u)) \preceq \text{approp}(\mathcal{S}(u), f)$

Definition 2.4 (Variable Assignment) *Let* $\mathcal{I} = \langle \mathcal{U}, \mathcal{S}, \mathcal{A} \rangle$ *be an interpretation and* VAR *a (countably infinite) set of variables. Variable assignment (denoted* $\alpha : \mathsf{VAR} \to \mathcal{U}$*) is a total function from the set of variables to the set of objects and assigns each variable in* VAR *an object in* U*.* $\text{ASS} = \mathcal{U}^{\mathsf{VAR}}$ *is the set of variable assignments in* $\mathcal{I}$*.*

[2]We will use the symbols "$\downarrow$" for "is defined" and "$\uparrow$" for "is not defined".

Definition 2.5 ($[\![\cdot]\!]^{\mathcal{I}}_{\alpha}$) *Term interpretation is defined with respect to an interpretation $\mathcal{I}$ and a variable assignment α:*

- $[\![x]\!]^{\mathcal{I}}_{\alpha} = \{\alpha(x)\}$ *if* $x \in \mathsf{VAR}$.
- $[\![t]\!]^{\mathcal{I}}_{\alpha} = \{u \in \mathcal{U} \mid \mathcal{S}(u) \preceq t\}$ *if* $t \in \mathcal{T}$.
- $[\![f : \phi]\!]^{\mathcal{I}}_{\alpha} = \{u \in \mathcal{U} \mid \mathcal{A}(f)(u) \downarrow \wedge \mathcal{A}(f)(u) \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}\}$ *if* $f \in \mathcal{F}$ *and* ϕ *is a term.*
- $[\![\phi_1 \wedge \phi_2]\!]^{\mathcal{I}}_{\alpha} = [\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \cap [\![\phi_2]\!]^{\mathcal{I}}_{\alpha}$, $[\![\phi_1 \vee \phi_2]\!]^{\mathcal{I}}_{\alpha} = [\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \cup [\![\phi_2]\!]^{\mathcal{I}}_{\alpha}$,
  $[\![\neg\phi_1]\!]^{\mathcal{I}}_{\alpha} = \mathcal{U} \setminus [\![\phi_1]\!]^{\mathcal{I}}_{\alpha}$ *if* ϕ_1 *and* ϕ_2 *are terms.*

We can now abstract away from the variable assignment.

Definition 2.6 ($[\![\cdot]\!]^{\mathcal{I}}$) $[\![\phi]\!]^{\mathcal{I}} = \bigcup_{\alpha \in \mathrm{ASS}} [\![\phi]\!]^{\mathcal{I}}_{\alpha}$

3 Set-valued Features

Following (Manandhar 1995b) and (Manandhar 1993), we will add set-valued attributes to our typed feature logic. One purpose of a feature logic with set-valued attributes is to provide an approach to model *linear precedence* (LP) rules within a linguistic theory like HPSG.[3] The main difference between "ordinary" and set-valued features is that the latter can occur more than once. So, if r is a set-valued attribute, f is a feature, ϕ and ψ are feature terms, then $f : \phi \wedge f : \psi = f : (\phi \wedge \psi)$, but $r : \phi \wedge r : \psi \neq r : (\phi \wedge \psi)$.

Manandhar (1995b) not only adds set-valued attributes to his logic, but also supplies additional descriptions to express the transitive and the reflexive-transitive closure of set-valued attributes, "first daughter", domain precedence and subset inclusion. In contrast to the logic we have introduced in the second section, the feature logic in (Manandhar 1995b) is untyped and lacks the possibility to express negation.[4]

In this section, I will extend the signature by set-valued attributes, the set of feature terms by universal and existential quantification over set-valued attributes, and show how set-valued attributes and feature terms containing them are interpreted. How the descriptions of (Manandhar 1995b) could be added, will be the topic of section 5.

[3] See (Pollard and Sag 1987).

[4] (Manandhar 1995a) presents another variant of an untyped feature logic with set-valued attributes in order to express relations between sets and members of sets. Universal quantification over complex feature terms and negation is not discussed in that approach.

Definition 3.1 (Signature) *A signature is a quintuple* $\langle \mathcal{T}, \preceq, \mathcal{F}, \mathcal{R}, \text{approp} \rangle$ *s.t.*

- $\mathcal{T}$ *is a finite set of types, and* $\langle \mathcal{T}, \preceq \rangle$ *is a join semi-lattice*
- $\mathcal{V} = \{t \in \mathcal{T} \mid \text{if } t' \preceq t \text{ then } t' = t\}$ *is the set of minimal types*
- $\mathcal{F}$ *is a finite set of feature names*
- $\mathcal{R}$ *is a finite set of set-valued feature (or attribute) names*
- $\text{approp} : \mathcal{V} \times (\mathcal{F} \cup \mathcal{R}) \rightharpoonup \mathcal{T}$ *is a partial function from pairs of minimal types and (set-valued) features to types*

Also, VAR *is a countably infinite set of variables.*

Henceforth, I will assume a fixed signature.

Definition 3.2 (Feature Terms) *The set of (feature) terms is the smallest set s.t.*

- x *is a term* *if* $x \in \mathsf{VAR}$.
- t *is a term* *if* $t \in \mathcal{T}$.
- $f : \phi$ *is a term* *if* $f \in \mathcal{F}$ *and* ϕ *is a term.*
- $r : \{\phi_1, ..., \phi_n\}$ *is a term* *if* $r \in \mathcal{R}$, $n \geq 1$ *and* $\phi_1, ..., \phi_n$ *are terms. If* $n = 1$, *we will also write* $r : \{\phi_1\}$ *instead of* $r : \phi_1$.
- $\forall r : \phi$ *is a term* *if* $r \in \mathcal{R}$ *and* ϕ *is a term.*
- $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg \phi_1$ *are terms* *if* ϕ_1 *and* ϕ_2 *are terms.*

The simplification of omitting the brackets in $r : \{\phi_1\}$ requires $\mathcal{F}$ and $\mathcal{R}$ to be disjoint sets. Otherwise, we cannot decide whether r in the feature term $r : \phi$ is a set-valued attribute with the singleton value $\{\phi\}$ or a feature with the value ϕ.

Definition 3.3 (Interpretation) *An interpretation is a quadruple* $\mathcal{I} = \langle \mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{D} \rangle$ *s.t.*

- $\mathcal{U}$ *is a set of objects, the domain of* $\mathcal{I}$
- $\mathcal{S} : \mathcal{U} \to \mathcal{V}$ *is a total function from the set of objects to the set of minimal types*
- $\mathcal{A} : \mathcal{F} \to \mathcal{U}^{\mathcal{U}}$ *is an attribute interpretation function s.t.*

- *for each* $u \in \mathcal{U}$, *for each* $f \in \mathcal{F}$, *for each* $t \in \mathcal{T}$, *if* $\text{approp}(\mathcal{S}(u), f)$ *is defined and* $\text{approp}(\mathcal{S}(u), f) = t$, *then* $\mathcal{A}(f)(u)$ *is defined and* $\mathcal{S}(\mathcal{A}(f)(u)) \preceq t$
- *for each* $u \in \mathcal{U}$, *for each* $f \in \mathcal{F}$, *if* $\mathcal{A}(f)(u)$ *is defined, then* $\text{approp}(\mathcal{S}(u), f)$ *is defined and* $\mathcal{S}(\mathcal{A}(f)(u)) \preceq \text{approp}(\mathcal{S}(u), f)$

- $\mathcal{D} : \mathcal{R} \rightarrow \mathcal{P}(\mathcal{U})^{\mathcal{U}}$ *is a set-attribute interpretation function s.t.*
  - *for each* $u \in \mathcal{U}$, *for each* $r \in \mathcal{R}$, *for each* $t \in \mathcal{T}$, *if* $\text{approp}(\mathcal{S}(u), r)$ *is defined and* $\text{approp}(\mathcal{S}(u), r) = t$, *then for each* $u' \in \mathcal{D}(r)(u), \mathcal{S}(u') \preceq t$
  - *for each* $u \in \mathcal{U}$, *for each* $r \in \mathcal{R}$, *if* $\mathcal{D}(r)(u) \neq \emptyset$, *then* $\text{approp}(\mathcal{S}(u), r)$ *is defined and for each* $u' \in \mathcal{D}(r)(u)$, $\mathcal{S}(u') \preceq \text{approp}(\mathcal{S}(u), r)$

Again, $\text{ASS} = \mathcal{U}^{\mathsf{VAR}}$ *is the set of variable assignments in* $\mathcal{I}$.

Definition 3.4 ($[\![\cdot]\!]^{\mathcal{I}}_{\alpha}$) *Term interpretation is defined with respect to an interpretation* $\mathcal{I}$ *and a variable assignment* α:

- $[\![x]\!]^{\mathcal{I}}_{\alpha} = \{\alpha(x)\}$ *if* $x \in \mathsf{VAR}$.
- $[\![t]\!]^{\mathcal{I}}_{\alpha} = \{u \in \mathcal{U} \mid \mathcal{S}(u) \preceq t\}$ *if* $t \in \mathcal{T}$.
- $[\![f : \phi]\!]^{\mathcal{I}}_{\alpha} = \{u \in \mathcal{U} \mid \mathcal{A}(f)(u) \downarrow \wedge \mathcal{A}(f)(u) \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}\}$ *if* $f \in \mathcal{F}$ *and* ϕ *is a term.*
- $[\![r : \{\phi_1, ...\phi_n\}]\!]^{\mathcal{I}}_{\alpha} = \{u \in \mathcal{U} \mid \exists u_1, ..., u_n : u_1, ..., u_n \in \mathcal{D}(r)(u) \wedge u_1 \in [\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \wedge ... \wedge u_n \in [\![\phi_n]\!]^{\mathcal{I}}_{\alpha}\}$ *if* $r \in \mathcal{R}$ *and each* $\phi_i \in \phi_1, ...\phi_n\}$ *is a term.*
- $[\![\forall r : \phi]\!]^{\mathcal{I}}_{\alpha} = \{u \in \mathcal{U} \mid \forall u' : u' \in \mathcal{D}(r)(u) \rightarrow u' \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}\}$ *if* $r \in \mathcal{R}$ *and* ϕ *is a term.*
- $[\![\phi_1 \wedge \phi_2]\!]^{\mathcal{I}}_{\alpha} = [\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \cap [\![\phi_2]\!]^{\mathcal{I}}_{\alpha}$, $[\![\phi_1 \vee \phi_2]\!]^{\mathcal{I}}_{\alpha} = [\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \cup [\![\phi_2]\!]^{\mathcal{I}}_{\alpha}$, $[\![\neg\phi_1]\!]^{\mathcal{I}}_{\alpha} = \mathcal{U} \setminus [\![\phi_1]\!]^{\mathcal{I}}_{\alpha}$ *if* ϕ_1 *and* ϕ_2 *are terms.*

Note, in particular, that unlike the attribute interpretation function for features, the set-attribute interpretation function does not need to be defined. The existential quantifier in the denotation of $r : \{\phi_1, ...\phi_n\}$ ensures that $\mathcal{D}(r)(u)$ is not empty. Clearly, the same is true for $r^+ : \{\phi_1, ...\phi_n\}$.[5] As far as the universal quantifier in the feature term $\forall r : \phi$ (and in $\forall r^+ : \phi$) is concerned, $\mathcal{D}(r)(u)$ might possibly be empty.

[5]We will define transitive closure in section 4.3.

But this is exactly what we want. If an object u has no set-valued feature r (i.e., $\mathcal{D}(r)(u)$ is empty), we want it to be in the denotation of $\forall r : \phi$ as well. Since $\mathcal{D}(r)(u) = \emptyset$ implies $\mathcal{D}(r^+)(u) = \emptyset$, we will see that $\forall r^+ : \phi$ has the same property.

As in definition 2.6, we can now abstract away from the variable assignment.

Definition 3.5 ($[\![\cdot]\!]^{\mathcal{I}}$) $[\![\phi]\!]^{\mathcal{I}} = \bigcup\limits_{\alpha \in \mathrm{ASS}} [\![\phi]\!]^{\mathcal{I}}_{\alpha}$

Proposition 3.1 *Let $S = \langle \mathcal{T}, \preceq, \mathcal{F}, \mathcal{R}, approp \rangle$ be a signature. For every S-interpretation $\mathcal{I} = \langle \mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{D} \rangle$,*
$[\![r : \{\phi_1, ..., \phi_n\}]\!]^{\mathcal{I}} = [\![r : \phi_1 \wedge ... \wedge r : \phi_n]\!]^{\mathcal{I}}$.

Proof. $[\![r : \{\phi_1, ..., \phi_n\}]\!]^{\mathcal{I}}$

$$= \bigcup_{\alpha \in \mathrm{ASS}} [\![r : \{\phi_1, ..., \phi_n\}]\!]^{\mathcal{I}}_{\alpha}$$

$$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \exists u_1 ... u_n : u_1 ... u_n \in \mathcal{D}(r)(u) \wedge u_1 \in [\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \wedge ... \wedge u_n \in [\![\phi_n]\!]^{\mathcal{I}}_{\alpha}\}$$

$$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \exists u_1 \in \mathcal{D}(r)(u) \wedge u_1 \in [\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \wedge ... \wedge \exists u_n \in \mathcal{D}(r)(u) \wedge u_n \in [\![\phi_n]\!]^{\mathcal{I}}_{\alpha}\}$$

$$= \bigcup_{\alpha \in \mathrm{ASS}} (\{u \in \mathcal{U} \mid \exists u_1 : u_1 \in \mathcal{D}(r)(u) \wedge u_1 \in [\![\phi_1]\!]^{\mathcal{I}}_{\alpha}\} \cap ... \cap \{u \in \mathcal{U} \mid \exists u_n : u_n \in \mathcal{D}(r)(u) \wedge u_n \in [\![\phi_n]\!]^{\mathcal{I}}_{\alpha}\})$$

$$= \bigcup_{\alpha \in \mathrm{ASS}} ([\![r : \{\phi_1\}]\!]^{\mathcal{I}}_{\alpha} \cap ... \cap [\![r : \{\phi_n\}]\!]^{\mathcal{I}}_{\alpha})$$

$$= \bigcup_{\alpha \in \mathrm{ASS}} [\![r : \{\phi_1\} \wedge ... \wedge r : \{\phi_n\}]\!]^{\mathcal{I}}_{\alpha}$$

$$= \bigcup_{\alpha \in \mathrm{ASS}} [\![r : \phi_1 \wedge ... \wedge r : \phi_n]\!]^{\mathcal{I}}_{\alpha}$$

$$= [\![r : \phi_1 \wedge ... \wedge r : \phi_n]\!]^{\mathcal{I}}$$

□

4 Constraint Solving

Given the extended feature logic with set-valued attributes, we need a procedure to verify the satisfiability of a feature term.

Constraint solving[6] is a technique to determine the satisfiability of a given feature term. The main idea is to try to bring a set of constraints that is equivalent to the original feature term into some normal form.

If a normal is obtained, we know that the term was satisfiable, otherwise it was not. In both cases the procedure terminates. Following Smolka (1988) and Götz (prep), we start by transforming the feature term into disjunctive normal form without negation (DNF) except in front of variables. Then, another procedure translates the term in DNF into an equivalent set of feature constraints, and finally, the satisfiability of this set is checked by a rewrite system which decides whether it is possible to bring the set into (satisfiable) normal form or not.

4.1 Disjunctive Normal Form

If the set of objects denoted by a feature term is non-empty, then the term is satisfiable (and vice versa):

Definition 4.1 *A feature term ϕ is satisfiable iff there exists an interpretation $\mathcal{I}$ s.t. $[\![\phi]\!]^{\mathcal{I}} \neq \emptyset$.*

The first step to check this satisfiability is to bring a feature term into disjunctive normal form (DNF). We can use the following equations to transform an arbitrary feature term in a finite number of steps into an equivalent term where negation only occurs in front of variables and then apply three distributive laws.

The 6th of the following equations shows the interaction between set-valued attributes and the universal quantifier: Universal quantification automatically arises if we allow feature terms containing set-valued attributes to be negated. Since we know, by proposition 3.1, that
$[\![r : \{\phi_1, ..., \phi_n\}]\!]^{\mathcal{I}} = [\![r : \phi_1 \wedge ... \wedge r : \phi_n]\!]^{\mathcal{I}}$,
we will only consider set-valued features with *one* value.

Proposition 4.1 *For any feature term ϕ, for any feature term ψ, for any type $t \in \mathcal{T}$, for any feature f, for any set-valued feature r, for any type t, the following equivalences hold:*

1. $\neg\neg\phi \equiv \phi$
2. $\neg(\phi \wedge \psi) \equiv (\neg\phi \vee \neg\psi)$

[6] For further reading about Constraint Logic Programming (CLP) see (Jaffar and Lassez 1987) or (Höhfeld and Smolka 1988).

3. $\neg(\phi \vee \psi) \equiv (\neg\phi \wedge \neg\psi)$

4. $\neg t \equiv \bigvee\{t' \mid t' \in \mathcal{V} \wedge t' \not\preceq t\}$

5. $\neg(f : \phi) \equiv f : \neg\phi \vee \bigvee\{t \mid approp(t, f) \uparrow\}$

6. $\neg(r : \phi) \equiv \forall r : \neg\phi$

7. $\neg(\forall r : \phi) \equiv r : \neg\phi$

Proof. Let $S = \langle \mathcal{T}, \preceq, \mathcal{F}, \mathcal{R}, approp \rangle$ be a signature. For every S-interpretation $\mathcal{I} = \langle \mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{D} \rangle$,

1. for any term ϕ,
$[\![\neg\neg\phi]\!]^{\mathcal{I}}$
$= \bigcup\limits_{\alpha\in\text{ASS}} [\![\neg\neg\phi]\!]^{\mathcal{I}}_{\alpha}$
$= \bigcup\limits_{\alpha\in\text{ASS}} \mathcal{U}\backslash(\mathcal{U}\backslash[\![\phi]\!]^{\mathcal{I}}_{\alpha})$
$= \bigcup\limits_{\alpha\in\text{ASS}} [\![\phi]\!]^{\mathcal{I}}_{\alpha}$
$= [\![\phi]\!]^{\mathcal{I}}$,

2. for any term ϕ, for any term ψ,
$[\![\neg(\phi \wedge \psi)]\!]^{\mathcal{I}}$
$= \bigcup\limits_{\alpha\in\text{ASS}} [\![\neg(\phi \wedge \psi)]\!]^{\mathcal{I}}_{\alpha}$
$= \bigcup\limits_{\alpha\in\text{ASS}} \mathcal{U}\backslash[\![\phi \wedge \psi]\!]^{\mathcal{I}}_{\alpha}$
$= \bigcup\limits_{\alpha\in\text{ASS}} \mathcal{U}\backslash([\![\phi]\!]^{\mathcal{I}}_{\alpha} \cap [\![\psi]\!]^{\mathcal{I}}_{\alpha})$
$= \bigcup\limits_{\alpha\in\text{ASS}} \mathcal{U}\backslash[\![\phi]\!]^{\mathcal{I}}_{\alpha} \cup \mathcal{U}\backslash[\![\psi]\!]^{\mathcal{I}}_{\alpha}$
$= \bigcup\limits_{\alpha\in\text{ASS}} [\![\neg\phi \vee \neg\psi]\!]^{\mathcal{I}}_{\alpha}$
$= [\![\neg\phi \vee \neg\psi]\!]^{\mathcal{I}}$,

3. for any term ϕ, for any term ψ,
$[\![\neg(\phi \vee \psi)]\!]^{\mathcal{I}}$
$= \bigcup\limits_{\alpha\in\text{ASS}} [\![\neg(\phi \vee \psi)]\!]^{\mathcal{I}}_{\alpha}$
$= \bigcup\limits_{\alpha\in\text{ASS}} \mathcal{U}\backslash[\![\phi \vee \psi]\!]^{\mathcal{I}}_{\alpha}$
$= \bigcup\limits_{\alpha\in\text{ASS}} \mathcal{U}\backslash([\![\phi]\!]^{\mathcal{I}}_{\alpha} \cup [\![\psi]\!]^{\mathcal{I}}_{\alpha})$
$= \bigcup\limits_{\alpha\in\text{ASS}} \mathcal{U}\backslash[\![\phi]\!]^{\mathcal{I}}_{\alpha} \cap \mathcal{U}\backslash[\![\psi]\!]^{\mathcal{I}}_{\alpha}$
$= \bigcup\limits_{\alpha\in\text{ASS}} [\![\neg\phi \wedge \neg\psi]\!]^{\mathcal{I}}_{\alpha}$
$= [\![\neg\phi \wedge \neg\psi]\!]^{\mathcal{I}}$,

4. for any type t,
$\llbracket \neg t \rrbracket^{\mathcal{I}}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \llbracket \neg t \rrbracket^{\mathcal{I}}_{\alpha}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \mathcal{U} \backslash \llbracket t \rrbracket^{\mathcal{I}}_{\alpha}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \mathcal{U} \backslash \{u \in \mathcal{U} \mid \mathcal{S}(u) \preceq t\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \mathcal{S}(u) \not\preceq t\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \llbracket \bigvee \{t' \mid t' \in \mathcal{V} \wedge t' \not\preceq t\} \rrbracket^{\mathcal{I}}_{\alpha}$
$= \llbracket \bigvee \{t' \mid t' \in \mathcal{V} \wedge t' \not\preceq t\} \rrbracket^{\mathcal{I}}$,

5. for any feature f, for any term ϕ,
$\llbracket \neg(f : \phi) \rrbracket^{\mathcal{I}}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \llbracket \neg(f : \phi) \rrbracket^{\mathcal{I}}_{\alpha}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \mathcal{U} \backslash \llbracket f : \phi \rrbracket^{\mathcal{I}}_{\alpha}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \mathcal{U} \backslash \{u \in \mathcal{U} \mid \mathcal{A}(f)(u) \downarrow \wedge \mathcal{A}(f)(u) \in \llbracket \phi \rrbracket^{\mathcal{I}}_{\alpha}\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} (\{u \in \mathcal{U} \mid \mathcal{A}(f)(u) \uparrow\} \cup$
$\{u \in \mathcal{U} \mid \mathcal{A}(f)(u) \notin \llbracket \phi \rrbracket^{\mathcal{I}}_{\alpha}\})$
$= \bigcup_{\alpha \in \mathrm{ASS}} (\{u \in \mathcal{U} \mid \mathit{approp}(\mathcal{S}(u), f) \uparrow\} \cup$
$\{u \in \mathcal{U} \mid \mathcal{A}(f)(u) \in \llbracket \neg\phi \rrbracket^{\mathcal{I}}_{\alpha}\})$
$= \bigcup_{\alpha \in \mathrm{ASS}} (\bigcup \{\llbracket t \rrbracket^{\mathcal{I}}_{\alpha} \mid \mathit{approp}(t, f) \uparrow\} \cup \llbracket f : \neg\phi \rrbracket^{\mathcal{I}}_{\alpha})$
$= \bigcup_{\alpha \in \mathrm{ASS}} \llbracket f : \neg\phi \vee \bigvee \{t \mid \mathit{approp}(t, f) \uparrow\} \rrbracket^{\mathcal{I}}_{\alpha}$
$= \llbracket f : \neg\phi \vee \bigvee \{t \mid \mathit{approp}(t, f) \uparrow\} \rrbracket^{\mathcal{I}}$,

6. for any set-valued feature r, for any term ϕ,
$\llbracket \neg(r : \phi) \rrbracket^{\mathcal{I}}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \llbracket \neg(r : \phi) \rrbracket^{\mathcal{I}}_{\alpha}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \mathcal{U} \backslash \{u \in \mathcal{U} \mid \exists u' : u' \in \mathcal{D}(r)(u) \wedge u' \in \llbracket \phi \rrbracket^{\mathcal{I}}_{\alpha}\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \neg(\exists u' : u' \in \mathcal{D}(r)(u) \wedge u' \in \llbracket \phi \rrbracket^{\mathcal{I}}_{\alpha})\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \forall u' : \neg(u' \in \mathcal{D}(r)(u) \wedge u' \in \llbracket \phi \rrbracket^{\mathcal{I}}_{\alpha})\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \forall u' : \neg(u' \in \mathcal{D}(r)(u)) \vee \neg(u' \in \llbracket \phi \rrbracket^{\mathcal{I}}_{\alpha})\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \forall u' : u' \in \mathcal{D}(r)(u) \rightarrow u' \notin \llbracket \phi \rrbracket^{\mathcal{I}}_{\alpha}\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \forall u' : u' \in \mathcal{D}(r)(u) \rightarrow u' \in \llbracket \neg\phi \rrbracket^{\mathcal{I}}_{\alpha}\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \llbracket \forall r : \neg\phi \rrbracket^{\mathcal{I}}_{\alpha}$

$= [\![\forall r : \neg\phi]\!]^{\mathcal{I}}$,

7. for any set-valued feature r, for any term ϕ,
$[\![\neg(\forall r : \phi)]\!]^{\mathcal{I}}$
$= \bigcup_{\alpha \in \mathrm{ASS}} [\![\neg(\forall r : \phi)]\!]^{\mathcal{I}}_{\alpha}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \mathcal{U} \backslash \{u \in \mathcal{U} \mid \forall u' : u' \in \mathcal{D}(r)(u) \to u' \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \neg(\forall u' : u' \in \mathcal{D}(r)(u) \to u' \in [\![\phi]\!]^{\mathcal{I}}_{\alpha})\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \exists u' : u' \in \mathcal{D}(r)(u) \wedge u' \notin [\![\phi]\!]^{\mathcal{I}}_{\alpha}\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} \{u \in \mathcal{U} \mid \exists u' : u' \in \mathcal{D}(r)(u) \wedge u' \in [\![\neg\phi]\!]^{\mathcal{I}}_{\alpha}\}$
$= \bigcup_{\alpha \in \mathrm{ASS}} [\![r : \neg\phi]\!]^{\mathcal{I}}_{\alpha}$
$= [\![r : \neg\phi]\!]^{\mathcal{I}}$

□

As discussed after definition 3.4, one of the differences between features and set-valued attributes is that we have no additional condition in the term interpretation of $r : \phi$ which states that for each member of $[\![r : \phi]\!]^{\mathcal{I}}_{\alpha}$, $\mathcal{D}(r)(u)$ has to be defined.

The consequence for negated feature terms is that the disjunction of all types not appropriate for a set-valued feature is missing after eliminating the negation in $\neg(r : \phi)$. Nevertheless, objects with no set-valued feature r defined are in the denotation of $\neg(r : \phi)$, since this follows immediately from the interpretation of $\forall r : \phi$ (or $\forall r : \neg\phi$).

Using the following additional transformations, we can now bring a feature term without negation except in front of variables into disjunctive normal form:

Proposition 4.2 *For each feature term ϕ, for each feature term ψ , for each feature term ξ , for each feature f, for each set-valued attribute r, the following equivalences hold:*

- $\phi \wedge (\psi \vee \xi) \equiv (\phi \wedge \psi) \vee (\phi \wedge \xi)$
- $f : (\phi \vee \psi) \equiv (f : \phi) \vee (f : \psi)$
- $r : (\phi \vee \psi) \equiv (r : \phi) \vee (r : \psi)$

Proof. The first (distributive) law is obvious, but the second and the third might not be trivial:

1. for any feature f, for any term ϕ, for any term ψ,
$[\![f : (\phi \vee \psi)]\!]^{\mathcal{I}}$
$= \bigcup_{\alpha \in \mathrm{ASS}} [\![f : (\phi \vee \psi)]\!]^{\mathcal{I}}_{\alpha}$

$$
\begin{aligned}
&= \bigcup_{\alpha\in\mathrm{ASS}} \{u \in \mathcal{U} \mid \mathcal{A}(f)(u)\downarrow \wedge \mathcal{A}(f)(u) \in [\![\phi \vee \psi]\!]^{\mathcal{I}}_{\alpha}\} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} \{u \in \mathcal{U} \mid \mathcal{A}(f)(u)\downarrow \wedge \mathcal{A}(f)(u) \in ([\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \cup [\![\phi_2]\!]^{\mathcal{I}}_{\alpha})\} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} \{u \in \mathcal{U} \mid \mathcal{A}(f)(u)\downarrow \wedge (\mathcal{A}(f)(u) \in [\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \vee \mathcal{A}(f)(u) \in [\![\phi_2]\!]^{\mathcal{I}}_{\alpha})\} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} \{u \in \mathcal{U} \mid (\mathcal{A}(f)(u)\downarrow \wedge \mathcal{A}(f)(u) \in [\![\phi_1]\!]^{\mathcal{I}}_{\alpha}) \vee \\
&\qquad (\mathcal{A}(f)(u)\downarrow \wedge \mathcal{A}(f)(u) \in [\![\phi_2]\!]^{\mathcal{I}}_{\alpha})\} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} (\{u \in \mathcal{U} \mid \mathcal{A}(f)(u)\downarrow \wedge \mathcal{A}(f)(u) \in [\![\phi_1]\!]^{\mathcal{I}}_{\alpha}\} \cup \\
&\qquad \{u \in \mathcal{U} \mid \mathcal{A}(f)(u)\downarrow \wedge \mathcal{A}(f)(u) \in [\![\phi_2]\!]^{\mathcal{I}}_{\alpha}\}) \\
&= \bigcup_{\alpha\in\mathrm{ASS}} [\![(f : \phi) \vee (f : \psi)]\!]^{\mathcal{I}}_{\alpha} \\
&= [\![(f : \phi) \vee (f : \psi)]\!]^{\mathcal{I}}
\end{aligned}
$$

2. For any set-valued attribute r, for any term ϕ, for any term ψ,

$$
\begin{aligned}
&[\![r : (\phi \vee \psi)]\!]^{\mathcal{I}} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} [\![r : (\phi \vee \psi)]\!]^{\mathcal{I}}_{\alpha} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} \{u \in \mathcal{U} \mid \exists u' : u' \in \mathcal{D}(r)(u) \wedge u' \in [\![\phi \vee \psi]\!]^{\mathcal{I}}_{\alpha}\} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} \{u \in \mathcal{U} \mid \exists u' : u' \in \mathcal{D}(r)(u) \wedge (u' \in [\![\phi]\!]^{\mathcal{I}}_{\alpha} \cup [\![\psi]\!]^{\mathcal{I}}_{\alpha})\} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} \{u \in \mathcal{U} \mid \exists u' : u' \in \mathcal{D}(r)(u) \wedge (u' \in [\![\phi]\!]^{\mathcal{I}}_{\alpha} \vee u' \in [\![\psi]\!]^{\mathcal{I}}_{\alpha})\} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} \{u \in \mathcal{U} \mid \exists u' : (u' \in \mathcal{D}(r)(u) \wedge u' \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}) \vee \\
&\qquad (u' \in \mathcal{D}(r)(u) \wedge u' \in [\![\psi]\!]^{\mathcal{I}}_{\alpha})\} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} \{u \in \mathcal{U} \mid (\exists u' : u' \in \mathcal{D}(r)(u) \wedge u' \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}) \vee \\
&\qquad (\exists u' : u' \in \mathcal{D}(r)(u) \wedge u' \in [\![\psi]\!]^{\mathcal{I}}_{\alpha})\} \\
&= \bigcup_{\alpha\in\mathrm{ASS}} (\{u \in \mathcal{U} \mid \exists u' : u' \in \mathcal{D}(r)(u) \wedge u' \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}\} \cup \\
&\qquad \{u \in \mathcal{U} \mid \exists u' : u' \in \mathcal{D}(r)(u) \wedge u' \in [\![\psi]\!]^{\mathcal{I}}_{\alpha}\}) \\
&= \bigcup_{\alpha\in\mathrm{ASS}} [\![(r : \phi) \vee (r : \psi)]\!]^{\mathcal{I}}_{\alpha} \\
&= [\![(r : \phi) \vee (r : \psi)]\!]^{\mathcal{I}}
\end{aligned}
$$

□

Unfortunately, these two equivalences have no correspondence for feature terms with set-valued attributes under the scope of a universal quantifier, i.e., a feature term of the form $\forall r : (\phi \vee \psi)$ cannot be transformed into $(\forall r : \phi) \vee (\forall r : \psi)$. This can easily be seen in the second part of the proof: The crucial step there to bring the disjunction to the top level[7] cannot be made with an universal instead of the existential quantifier.

[7] This rule can be derived in a deduction calculus for first order logic, see for example (Ebbinghaus et al. 1992).

The proof is easy since one can construct a model in which $\forall r : (\phi \vee \psi)$ is true, but $(\forall r : \phi) \vee (\forall r : \psi)$ is false.

But this means that the disjunctive normal form described above in propositions 4.1 and 4.2 does contain disjunction (under the scope of a quantifier). For the sake of simplicity, we will assume that there is no disjunction under the scope of the universal quantifier and will provide a possible approach to solve that problem in section 4.3.

4.2 Feature Constraints

A feature term in DNF can be split up into atomic parts which Smolka (1988) calls *feature constraints*. Each disjunct of the feature term is represented by a set of feature constraints, which are interpreted conjunctively. The idea is to bring this set of constraints into some normal form that is satisfiable. Sets of constraints and hence feature terms without negation (except in front of variables) are interpreted disjunctively. We will see this in section 4.2.2 when we consider the satisfiability of sets (of sets) of constraints. If, while the rewrite rules are applied, a set of feature constraints turns out to be inconsistent, it is eliminated from the set of sets of constraints.

First, I will describe the properties of the normal form and then introduce a rewrite system to produce it.

4.2.1 From Feature Terms to Feature Clauses

In this section, I will define the translation function from feature terms to feature constraints. At first, we need to know what a feature constraint is.

In order to define a feature constraint, I will introduce the notion of a *big variable*[8] and therefore extend our variable assignment function α, which assigns to each variable $x \in \mathsf{VAR}$ an object $u \in \mathcal{U}$ and to each big variable $\boldsymbol{X}$ a set of objects $\mathcal{U}' \subseteq \mathcal{U}$. The members of VAR will be called *small* variables to distinguish them from the members of BV, the *big* variables.

Definition 4.2 (Free Variables) *Let ϕ be a feature term. Then $\mathsf{FV}(\phi)$ is the set of all small variables occurring in ϕ.*

Definition 4.3 (Big Variables) BV *is a countably infinite set, the set of big variables.*

We will use the notation $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$, ... for big variables. In contrast to the variables we know so far (the small variables), big variables denote subsets of the universe:

[8] See (Manandhar 1993).

Definition 4.4 (Variable Assignment) *Let* $\alpha_{\mathsf{VAR}} : \mathsf{VAR} \to \mathcal{U}$ *be a total function from the set of small variables to the universe and let* α_{BV} : $\mathsf{BV}\!\to 2^{\mathcal{U}}$ *be a total function from the set of big variables to the power set of the universe. And let* $\bar{\boldsymbol{X}}$ *range over* $\mathsf{VAR} \cup \mathsf{BV}$. *Then the variable assignment* α *is a total function* $\alpha : \mathsf{VAR}\cup\mathsf{BV}\!\to \mathcal{U} \cup \mathcal{P}(\mathcal{U})$ *and defined as follows:*

$$\alpha(\bar{\boldsymbol{X}}) = \begin{cases} \alpha_{\mathsf{VAR}}(\bar{\boldsymbol{X}}) & \textit{if } \bar{\boldsymbol{X}} \in \mathsf{VAR} \\ \alpha_{\mathsf{BV}}(\bar{\boldsymbol{X}}) & \textit{if } \bar{\boldsymbol{X}} \in \mathsf{BV} \end{cases}$$

Definition 4.5 (Feature Constraints) *A feature constraint is an expression of the form*

- $x \mid y$
- $\boldsymbol{X} \mid y$
- $x \mid \neg y$
- $\boldsymbol{X} \mid \neg y$
- $x \mid t$
- $\boldsymbol{X} \mid t$
- $x \mid f : y$
- $\boldsymbol{X} \mid f : \boldsymbol{Y}$
- $x \mid r : y$
- $\boldsymbol{X} \mid r : y$
- $x \mid \forall r : \boldsymbol{Y}$
- $\boldsymbol{X} \mid \forall r : \boldsymbol{Y}$
- $\boldsymbol{X} - y$

where x, y *are small variables,* $\boldsymbol{X}$ *and* $\boldsymbol{Y}$ *are big variables,* t *is a type symbol,* f *is a feature symbol, and* r *is a set-valued attribute.*

Now we are ready to describe the relation between feature terms and feature constraints. Each feature term (without disjunction) can be translated into a set of feature constraints, and a disjunction of feature terms corresponds to a set of sets of feature constraints. We will call a set of feature constraints a feature clause and a set of feature clauses a matrix.[9]

Translating a feature term into a feature clause is defined recursively and given below in terms of the translation function **trans** , which is based on the function **trans** defined in (Dörre and Dorna 1993).

[9]See (Smolka 1988).

Definition 4.6 (trans) *The translation of a feature term ϕ into a set of feature constraints starts with a* new *small variable x s.t. $x \notin \mathsf{FV}(\phi)$.*

1. $\mathbf{trans}(x, a) = \{x \mid a\}$, *if a is a type.*
2. $\mathbf{trans}(\boldsymbol{X}, a) = \{\boldsymbol{X} \mid a\}$, *if a is a type.*
3. $\mathbf{trans}(x, y) = \{x \mid y\}$, *if y is a small variable.*
4. $\mathbf{trans}(\boldsymbol{X}, y) = \{\boldsymbol{X} \mid y\}$, *if y is a small variable.*
5. $\mathbf{trans}(x, \neg y) = \{x \mid \neg y\}$, *if y is a small variable.*
6. $\mathbf{trans}(\boldsymbol{X}, \neg y) = \{\boldsymbol{X} \mid \neg y\}$, *if y is a small variable.*
7. $\mathbf{trans}(x, f : \phi) = \{x \mid f : y\} \cup \mathbf{trans}(y, \phi)$,
   where y is a new small variable, f is a feature symbol and ϕ is an arbitrary feature term without disjunction.
8. $\mathbf{trans}(\boldsymbol{X}, f : \phi) = \{\boldsymbol{X} \mid f : \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$,
   where $\boldsymbol{Y}$ is a new big variable, f is a feature symbol and ϕ is an arbitrary feature term without disjunction.
9. $\mathbf{trans}(x, r : \phi) = \{x \mid r : y\} \cup \mathbf{trans}(y, \phi)$,
   where y is a new small variable, r is a set-valued attribute and ϕ is an arbitrary feature term without disjunction.
10. $\mathbf{trans}(\boldsymbol{X}, r : \phi) = \{\boldsymbol{X} \mid r : y\} \cup \mathbf{trans}(y, \phi)$,
    where y is a new small variable, r is a set-valued attribute and ϕ is an arbitrary feature term without disjunction.
11. $\mathbf{trans}(x, \forall r : \phi) = \{x \mid \forall r : \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$,
    where $\boldsymbol{Y}$ is a new big variable, r is a set-valued attribute and ϕ is an arbitrary feature term without disjunction.
12. $\mathbf{trans}(\boldsymbol{X}, \forall r : \phi) = \{\boldsymbol{X} \mid \forall r : \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$,
    where $\boldsymbol{Y}$ is a new big variable, r is a set-valued attribute and ϕ is an arbitrary feature term without disjunction.
13. $\mathbf{trans}(x, \psi \wedge \phi) = \mathbf{trans}(x, \psi) \cup \mathbf{trans}(x, \phi)$,
    if ψ and ϕ are arbitrary feature terms without disjunction.
14. $\mathbf{trans}(\boldsymbol{X}, \psi \wedge \phi) = \mathbf{trans}(\boldsymbol{X}, \psi) \cup \mathbf{trans}(\boldsymbol{X}, \phi)$,
    if ψ and ϕ are arbitrary feature terms without disjunction.

If we compare definition 4.5 to the feature constraints presented in (Smolka 1988), we notice that some of the constraints defined above are new. In addition to the constraints $x \mid r : y$, $\boldsymbol{X} \mid r : y$, $x \mid \forall r : \boldsymbol{Y}$ and $\boldsymbol{X} \mid \forall r : \boldsymbol{Y}$ which arise from feature terms of the form $r : \phi$ and $\forall r : \phi$, respectively, $\boldsymbol{X} - y$ is a constraint that has no corresponding feature term from which it is produced. This might seem strange, but as we will see in the rewrite system presented in 4.2.3, we will use this kind of constraint as some "auxiliary" or "memory" constraint. Additionally, it could seem inconsequent that $\mathbf{trans}(\boldsymbol{X}, f : \phi)$ results in $\{x \mid f : \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$, but $\mathbf{trans}(\boldsymbol{X}, r : \phi) = \{\boldsymbol{X} \mid r : y\} \cup \mathbf{trans}(y, \phi)$, where f is a feature and r is a set-valued attribute. The reason for the different translation of feature terms with set-valued and normal attributes and the interaction with feature constraints of the form $\boldsymbol{X} - y$ will become clearer when the normal form and the rewrite system are presented.

The reader should keep in mind that we aren't able yet to translate an arbitrary feature term of the form $\forall r : \phi$, where ϕ contains disjunction.

4.2.2 Satisfiability

Having a feature term translated into a set of feature constraints, we can now define how a feature constraint is satisfied by an interpretation and a variable assignment.

Definition 4.7 *For each interpretation $\mathcal{I}$, for each variable assignment α, for each small variable x, for each small variable y, for each big variable $\boldsymbol{X}$, for each big variable $\boldsymbol{Y}$, for each type t, for each feature f, for each set-valued attribute r,*

- $\mathcal{I}, \alpha \models x \mid y$ *iff* $\alpha(x) = \alpha(y)$.
- $\mathcal{I}, \alpha \models x \mid \neg y$ *iff* $\alpha(x) \neq \alpha(y)$.
- $\mathcal{I}, \alpha \models x \mid t$ *iff* $\mathcal{S}(\alpha(x)) \preceq t$.
- $\mathcal{I}, \alpha \models x \mid f : y$ *iff* $\mathcal{A}(f)(\alpha(x))\downarrow$ *and* $\mathcal{A}(f)(\alpha(x)) = \alpha(y)$.
- $\mathcal{I}, \alpha \models x \mid r : y$ *iff* $\alpha(y) \in \mathcal{D}(r)(\alpha(x))$.
- $\mathcal{I}, \alpha \models x \mid \forall r : \boldsymbol{Y}$ *iff* $\alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(\alpha(x))$.
- $\mathcal{I}, \alpha \models \boldsymbol{X} - y$ *iff* $\alpha(y) \in \alpha(\boldsymbol{X})$.
- $\mathcal{I}, \alpha \models \boldsymbol{X} \mid y$ *iff for each* $u \in \alpha(\boldsymbol{X}), u = \alpha(y)$.
- $\mathcal{I}, \alpha \models \boldsymbol{X} \mid \neg y$ *iff for each* $u \in \alpha(\boldsymbol{X}), u \neq \alpha(y)$.
- $\mathcal{I}, \alpha \models \boldsymbol{X} \mid t$ *iff for each* $u \in \alpha(\boldsymbol{X}), \mathcal{S}(u) \preceq t$.

- $\mathcal{I}, \alpha \models \boldsymbol{X} \mid f : \boldsymbol{Y}$ *iff for each* $u \in \alpha(\boldsymbol{X}), \mathcal{A}(f)(u) \downarrow$ *and* $\alpha(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$.
- $\mathcal{I}, \alpha \models \boldsymbol{X} \mid r : y$ *iff for each* $u \in \alpha(\boldsymbol{X}), \alpha(y) \in \mathcal{D}(r)(u)$.
- $\mathcal{I}, \alpha \models \boldsymbol{X} \mid \forall r : \boldsymbol{Y}$ *iff for each* $u \in \alpha(\boldsymbol{X}), \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u)$.

Definition 4.8 *For any feature constraint* σ*,* σ *is satisfiable iff there exists an interpretation* $\mathcal{I}$ *and a variable assignment* α *s.t.* $\mathcal{I}, \alpha \models \sigma$.

Definition 4.9 *For each set of feature constraints* Σ*, for each interpretation* $\mathcal{I}$*, for each variable assignment* α*,* $\mathcal{I}, \alpha \models \Sigma$ *iff for each constraint* $\sigma \in \Sigma$*,* $\mathcal{I}, \alpha \models \sigma$.

Definition 4.10 *For each set of constraints* Σ*,* Σ *is satisfiable iff there exists an interpretation* $\mathcal{I}$ *and a variable assignment* α *such that for each constraint* $\sigma \in \Sigma$*,* $\mathcal{I}, \alpha \models \sigma$.

Each feature constraint beginning with a small variable x has a corresponding feature constraint with a big variable $\boldsymbol{X}$ on the left-hand side. The satisfaction conditions of the latter constraints resemble those of the former: Since big variables denote sets of objects, the conditions have to be true of each object in the denotation of a big variable.

One should not confuse the constraints $\boldsymbol{X} - y$ and $\boldsymbol{X} \mid y$. $\boldsymbol{X} - y$ should be read "y is constrained by $\boldsymbol{X}$". $\boldsymbol{X} \mid y$ means that each entity in $\alpha(\boldsymbol{X})$ equals $alpha(y)$ but, though $\boldsymbol{X} \mid y$ means (almost) equality, $\alpha(\boldsymbol{X})$ is not required to be non-empty.

In practice, we will use constraints of the form $\boldsymbol{X} - y$ to copy all information about $\boldsymbol{X}$ to y without making $\boldsymbol{X}$ and y equal. The following example should make clear in which cases we cannot use equality: The feature term $(\forall r : t) \wedge (r : t') \wedge (r : t'')$ s.t. $t' \preceq t$, $t'' \preceq t$, $t' \not\preceq t''$ and $t'' \not\preceq t'$ is translated into the set $\{x \mid \forall r : \boldsymbol{Z}, x \mid r : z', x \mid r : z'',$ $\boldsymbol{Z} \mid t, z' \mid t', z'' \mid t''\ \}$. Obviously, the feature term is satisfiable. There are two set-valued attributes r, and the universal quantifier requires a type t for the value of each r. If we made both $\boldsymbol{Z}$ and z' and $\boldsymbol{Z}$ and z'' equal, then, by transitivity, z' would equal z'' though $t' \not\preceq t''$ and $t'' \not\preceq t'$. Consequently, the set of feature constraints would become unsatisfiable. When we define a normal form for sets of feature constraints we will see how information is copied by means of constraints of the form $\boldsymbol{X} - y$ in order to avoid clashes as in the example above.

Now we have to ensure that the translation function **trans** preserves the denotation of a feature term. The proof of the following proposition is divided into two subparts. And in each part, since **trans** produces both small and big variables, we have again two further assumptions to prove. The translation of a feature term as a disjunct D_i begins with a new small variable x. Therefore, we only consider small variables in the proposition. This means that we actually prove a stronger result from which the proposition clearly follows.

Proposition 4.3 *Let $D = D_1 \vee ... \vee D_n$ be in DNF.*
For each D_i, $1 \leq i \leq n$, for each interpretation $\mathcal{I} = \langle \mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{D} \rangle$, if $x \notin \mathsf{FV}(D)$, then
$[\![D_i]\!]^{\mathcal{I}} = \{\alpha(x) \mid \mathcal{I}, \alpha \models \mathbf{trans}(x, D_i)\}$

Proof. $\supseteq$: If $\mathcal{I}, \alpha \models \mathbf{trans}(x, D_i)$, then $\alpha(x) \in [\![D_i]\!]^{\mathcal{I}}$,
if $\mathcal{I}, \alpha \models \mathbf{trans}(\boldsymbol{X}, D_i)$, then $\alpha(\boldsymbol{X}) \subseteq [\![D_i]\!]^{\mathcal{I}}$.
Proof by induction on the definition of **trans**.

- $\mathcal{I}, \alpha \models \mathbf{trans}(x, a)$

  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid a\}$

  $\Rightarrow \mathcal{S}(\alpha(x)) \preceq a$

  $\Rightarrow \alpha(x) \in [\![a]\!]^{\mathcal{I}}_{\alpha}$

- $\mathcal{I}, \alpha \models \mathbf{trans}(x, y)$

  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid y\}$

  $\Rightarrow \alpha(x) = \alpha(y)$

  $\Rightarrow \alpha(x) \in [\![y]\!]^{\mathcal{I}}_{\alpha}$

- $\mathcal{I}, \alpha \models \mathbf{trans}(x, \neg y)$

  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid \neg y\}$

  $\Rightarrow \alpha(x) \neq \alpha(y)$

  $\Rightarrow \alpha(x) \in [\![\neg y]\!]^{\mathcal{I}}_{\alpha}$

- $\mathcal{I}, \alpha \models \mathbf{trans}(x, f : \phi)$

  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid f : y\} \cup \mathbf{trans}(y, \phi)$

  $\Rightarrow \mathcal{A}(f)(\alpha(x)) \downarrow$ and $\mathcal{A}(f)(\alpha(x)) = \alpha(y)$ and, by induction, $\alpha(y) \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}$

  $\Rightarrow \alpha(x) \in [\![f : \phi]\!]^{\mathcal{I}}_{\alpha}$

- $\mathcal{I}, \alpha \models \mathbf{trans}(x, r : \phi)$

  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid r : y\} \cup \mathbf{trans}(y, \phi)$

  $\Rightarrow \alpha(y) \in \mathcal{D}(r)(\alpha(x))$ and, by induction, $\alpha(y) \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}$

  $\Rightarrow \alpha(x) \in [\![r : \phi]\!]^{\mathcal{I}}_{\alpha}$

- $\mathcal{I}, \alpha \models \textbf{trans}(x, \forall r : \phi)$
  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid \forall r : \boldsymbol{Y}\} \cup \textbf{trans}(\boldsymbol{Y}, \phi)$
  $\Rightarrow \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(\alpha(x))$ and, by induction, $\alpha(\boldsymbol{Y}) \subseteq [\![\phi]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(x) \in [\![\forall r : \phi]\!]^{\mathcal{I}}_{\alpha}$
- $\mathcal{I}, \alpha \models \textbf{trans}(x, \psi \wedge \phi)$
  $\Rightarrow \mathcal{I}, \alpha \models \textbf{trans}(x, \psi) \cup \textbf{trans}(x, \phi)$
  $\Rightarrow \alpha(x) \in [\![\psi]\!]^{\mathcal{I}}_{\alpha}$ and, by induction, $\alpha(x) \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(x) \in [\![\psi \wedge \phi]\!]^{\mathcal{I}}_{\alpha}$
- $\mathcal{I}, \alpha \models \textbf{trans}(\boldsymbol{X}, a)$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\}$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), \mathcal{S}(u) \preceq a$
  $\Rightarrow \alpha(\boldsymbol{X}) \subseteq [\![a]\!]^{\mathcal{I}}_{\alpha}$
- $\mathcal{I}, \alpha \models \textbf{trans}(\boldsymbol{X}, y)$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid y\}$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), u = \alpha(y)$
  $\Rightarrow \alpha(\boldsymbol{X}) = \emptyset$ or $\alpha(\boldsymbol{X}) = \{\alpha(y)\} = [\![y]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(\boldsymbol{X}) \subseteq [\![y]\!]^{\mathcal{I}}_{\alpha}$
- $\mathcal{I}, \alpha \models \textbf{trans}(\boldsymbol{X}, \neg y)$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid \neg y\}$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), u \neq \alpha(y)$
  $\Rightarrow \alpha(\boldsymbol{X}) \subseteq [\![\neg y]\!]^{\mathcal{I}}_{\alpha}$ (since $[\![\neg y]\!]^{\mathcal{I}}_{\alpha} = \mathcal{U} \setminus \{\alpha(y)\}$)
- $\mathcal{I}, \alpha \models \textbf{trans}(\boldsymbol{X}, f : \phi)$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid f : \boldsymbol{Y}\} \cup \textbf{trans}(\boldsymbol{Y}, \phi)$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), \mathcal{A}(f)(u)\downarrow$ and $\alpha(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$
  and, by induction, $\alpha(\boldsymbol{Y}) \subseteq [\![\phi]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(\boldsymbol{X}) \subseteq [\![f : \phi]\!]^{\mathcal{I}}_{\alpha}$
- $\mathcal{I}, \alpha \models \textbf{trans}(\boldsymbol{X}, r : \phi)$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid r : y\} \cup \textbf{trans}(y, \phi)$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), \alpha(y) \in \mathcal{D}(r)(u)$ and, by induction,
  $\alpha(y) \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(\boldsymbol{X}) \subseteq [\![r : \phi]\!]^{\mathcal{I}}_{\alpha}$

- $\mathcal{I}, \alpha \models \mathbf{trans}(\boldsymbol{X}, \forall r : \phi)$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid \forall r :\ \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u)$ and, by induction,
  $\alpha(\boldsymbol{Y}) \subseteq [\![\phi]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), u \in [\![\forall r : \phi]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(\boldsymbol{X}) \subseteq [\![\forall r : \phi]\!]^{\mathcal{I}}_{\alpha}$
- $\mathcal{I}, \alpha \models \mathbf{trans}(\boldsymbol{X}, \psi \wedge \phi)$
  $\Rightarrow \mathcal{I}, \alpha \models \mathbf{trans}(\boldsymbol{X}, \psi) \cup \mathbf{trans}(\boldsymbol{X}, \phi)$
  $\Rightarrow \alpha(\boldsymbol{X}) \subseteq [\![\psi]\!]^{\mathcal{I}}_{\alpha}$ and, by induction, $\alpha(\boldsymbol{X}) \subseteq [\![\phi]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(\boldsymbol{X}) \subseteq [\![\psi \wedge \phi]\!]^{\mathcal{I}}_{\alpha}$

Since $\alpha(x) \in [\![D_i]\!]^{\mathcal{I}}_{\alpha}$ and $\alpha(\boldsymbol{X}) \subseteq [\![D_i]\!]^{\mathcal{I}}_{\alpha}$, it follows that $\alpha(x) \in [\![D_i]\!]^{\mathcal{I}}$ and $\alpha(\boldsymbol{X}) \subseteq [\![D_i]\!]^{\mathcal{I}}$.

$\subseteq$: If $u \in [\![D_i]\!]^{\mathcal{I}}_{\alpha}$, where $\alpha(x) = u$, then there is an α' s.t. $\alpha'(x) = u$, for each $y \in \mathsf{FV}(D), \alpha'(y) = \alpha(y)$ and $\mathcal{I}, \alpha' \models \mathbf{trans}(x, D_i)$. If $\mathcal{U}' \subseteq [\![D_i]\!]^{\mathcal{I}}_{\alpha}$, where $\alpha(\boldsymbol{X}) = \mathcal{U}'$, then there is an α' s.t. $\alpha'(\boldsymbol{X}) = \mathcal{U}'$, for each $y \in \mathsf{FV}(D), \alpha'(y) = \alpha(y)$ and $\mathcal{I}, \alpha' \models \mathbf{trans}(\boldsymbol{X}, D_i)$.

- $u \in [\![a]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(x) = u$
  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid a\}$
  $\Rightarrow \mathcal{I}, \alpha \models \mathbf{trans}(x, a)$
- $u \in [\![y]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(x) = u$
  $\Rightarrow \alpha(y) = u = \alpha(x)$
  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid y\}$
  $\Rightarrow \mathcal{I}, \alpha \models \mathbf{trans}(x, y)$
- $u \in [\![\neg y]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(x) = u$
  $\Rightarrow \alpha(y) \neq u = \alpha(x)$
  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid \neg y\}$
  $\Rightarrow \mathcal{I}, \alpha \models \mathbf{trans}(x, \neg y)$
- $u \in [\![f : \phi]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(x) = u$ and $\mathbf{trans}(x, f : \phi) = \{x \mid f : y\} \cup$ $\mathbf{trans}(y, \phi)$
  $\Rightarrow \exists \beta.\ \beta(y) = \mathcal{A}(f)(u)$, for each $z \in \mathsf{FV}(D)$,
  $\beta(z) = \alpha(z)$ and $\mathcal{I}, \beta \models \mathbf{trans}(y, \phi)$

(by induction, since $y \notin \mathsf{FV}(\phi)$)

$\Rightarrow \mathcal{I}, \beta_{[x \mapsto u]} \models \{x \mid f : y\} \cup \mathbf{trans}(y, \phi)$

(since $x \notin \mathsf{FV}(\mathbf{trans}(y, \phi))$)

$\Rightarrow \mathcal{I}, \beta_{[x \mapsto u]} \models \mathbf{trans}(x, f : \phi)$

- $u \in [\![r : \phi]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(x) = u$ and $\mathbf{trans}(x, r : \phi) = \{x \mid r : y\} \cup \mathbf{trans}(y, \phi)$

  $\Rightarrow \exists \beta.\ \beta(y) \in \mathcal{D}(r)(u)$, for each $z \in \mathsf{FV}(D)$, $\beta(z) = \alpha(z)$ and $\mathcal{I}, \beta \models \mathbf{trans}(y, \phi)$

  (by induction, since $y \notin \mathsf{FV}(\phi)$)

  $\Rightarrow \mathcal{I}, \beta_{[x \mapsto u]} \models \{x \mid r : y\} \cup \mathbf{trans}(y, \phi)$

  (since $x \notin \mathsf{FV}(\mathbf{trans}(y, \phi))$)

  $\Rightarrow \mathcal{I}, \beta_{[x \mapsto u]} \models \mathbf{trans}(x, r : \phi)$

- $u \in [\![\forall r : \phi]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(x) = u$ and $\mathbf{trans}(x, \forall r : \phi) = \{x \mid \forall r :\ \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$

  $\Rightarrow\ \exists \beta.\ \beta(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u)$, for each $z \in \mathsf{FV}(D)$, $\beta(z) = \alpha(z)$ and $\mathcal{I}, \beta \models \mathbf{trans}(\boldsymbol{Y}, \phi)$

  (by induction, since $\boldsymbol{Y} \notin \mathsf{FV}(\phi)$)

  $\Rightarrow \mathcal{I}, \beta_{[x \mapsto u]} \models \{x \mid \forall r :\ \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$

  (since $x \notin \mathsf{FV}(\mathbf{trans}(\boldsymbol{Y}, \phi))$)

  $\Rightarrow \mathcal{I}, \beta_{[x \mapsto u]} \models \mathbf{trans}(x, \forall r : \phi)$

- $u \in [\![\psi \wedge \phi]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(x) = u$

  $\Rightarrow u \in [\![\psi]\!]^{\mathcal{I}}_{\alpha}$ and $u \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}$

  $\Rightarrow \exists \beta.\ \mathcal{I}, \beta \models \mathbf{trans}(x, \psi)$, $\beta(x) = u$ and for each $z \in \mathsf{FV}(D)$, $\beta(z) = \alpha(z)$, and $\exists \beta'.\ \mathcal{I}, \beta' \models \mathbf{trans}(x, \phi)$, $\beta'(x) = u$ and for each $z \in \mathsf{FV}(D)$, $\beta'(z) = \alpha(z)$

  (by induction)

  $\Rightarrow$ if $\alpha' = \beta_{[y_1 \mapsto \beta'(y_1), \ldots, y_n \mapsto \beta'(y_n), \boldsymbol{Z}_1 \mapsto \beta'(\boldsymbol{Z}_1), \ldots, \boldsymbol{Z}_n \mapsto \beta'(\boldsymbol{Z}_n)]}$, where $\{y_1, \ldots, y_n\} = \mathsf{FV}(\mathbf{trans}(x, \phi)) \backslash (\{x\} \cup \mathsf{FV}(D))$ and $\{\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_n\} = \mathsf{BV}(\mathbf{trans}(x, \phi))$ then $\mathcal{I}, \alpha' \models \mathbf{trans}(x, \psi \wedge \phi)$

  (since $(\ \mathsf{FV}(\mathbf{trans}(x, \psi)) \backslash (\{x\} \cup \mathsf{FV}(D))) \cap (\ \mathsf{FV}(\mathbf{trans}(x, \phi)) \backslash (\{x\} \cup \mathsf{FV}(D))) = \emptyset$ and for each $y \in \mathsf{FV}(D)$, $\beta(y) = \beta'(y)$)

- $\mathcal{U}' \subseteq [\![a]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(\boldsymbol{X}) = \mathcal{U}'$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), u \in [\![a]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), u \preceq a$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\}$
  $\Rightarrow \mathcal{I}, \alpha \models \mathbf{trans}(\boldsymbol{X}, a)$
- $\mathcal{U}' \subseteq [\![y]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(\boldsymbol{X}) = \mathcal{U}'$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), u \in [\![y]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(\boldsymbol{X}) = \{\alpha(y)\} = \{u\}$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid y\}$
  $\Rightarrow \mathcal{I}, \alpha \models \mathbf{trans}(\boldsymbol{X}, y)$
- $\mathcal{U}' \subseteq [\![\neg y]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(\boldsymbol{X}) = \mathcal{U}'$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), u \neq \alpha(y)$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid \neg y\}$
  $\Rightarrow \mathcal{I}, \alpha \models \mathbf{trans}(\boldsymbol{X}, \neg y)$
- $\mathcal{U}' \subseteq [\![f : \phi]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(\boldsymbol{X}) = \mathcal{U}'$ and $\mathbf{trans}(\boldsymbol{X}, f : \phi)$
  $= \{\boldsymbol{X} \mid f : \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$
  $\Rightarrow \exists \beta.\ \forall u \in \alpha(\boldsymbol{X}), \mathcal{A}(f)(u) \downarrow$ and $\beta(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$,
  for each $z \in \mathsf{FV}(D)$, $\beta(z) = \alpha(z)$ and $\mathcal{I}, \beta \models \mathbf{trans}(\boldsymbol{Y}, \phi)$
  (by induction, since $\boldsymbol{Y} \notin \mathsf{FV}(\phi)$
  $\Rightarrow \mathcal{I}, \beta_{[\boldsymbol{X} \mapsto \mathcal{U}']} \models \{\boldsymbol{X} \mid f : \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$
  (since $\boldsymbol{X} \notin \mathsf{FV}(\mathbf{trans}(y, \phi)))$
  $\Rightarrow \mathcal{I}, \beta_{[\boldsymbol{X} \mapsto \mathcal{U}']} \models \mathbf{trans}(\boldsymbol{X}, f : \phi)$
- $\mathcal{U}' \subseteq [\![r : \phi]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(\boldsymbol{X}) = \mathcal{U}'$ and $\mathbf{trans}(\boldsymbol{X}, r : \phi)$
  $= \{\boldsymbol{X} \mid r : y\} \cup \mathbf{trans}(y, \phi)$
  $\Rightarrow \exists \beta.$ for each $u \in \alpha(\boldsymbol{X}), \beta(y) \in \mathcal{D}(r)(u)$, for each $z \in \mathsf{FV}(D)$,
  $\beta(z) = \alpha(z)$ and $\mathcal{I}, \beta \models \mathbf{trans}(y, \phi)$
  (by induction, since $y \notin \mathsf{FV}(\phi)$)
  $\Rightarrow \mathcal{I}, \beta_{[\boldsymbol{X} \mapsto \mathcal{U}']} \models \{\boldsymbol{X} \mid r : y\} \cup \mathbf{trans}(y, \phi)$
  (since $\boldsymbol{X} \notin \mathsf{FV}(\mathbf{trans}(y, \phi)))$
  $\Rightarrow \mathcal{I}, \beta_{[\boldsymbol{X} \mapsto \mathcal{U}']} \models \mathbf{trans}(\boldsymbol{X}, r : \phi)$

- $\mathcal{U}' \subseteq [\![\forall r : \phi]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(\boldsymbol{X}) = \mathcal{U}'$ and $\mathbf{trans}(\boldsymbol{X}, \forall r : \phi)$
  $= \{\boldsymbol{X} \mid \forall r : \ \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$
  $\Rightarrow \exists \beta.$ for each $u \in \alpha(\boldsymbol{X}), \beta(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u)$, for each $z \in \mathsf{FV}(D)$,
  $\beta(z) = \alpha(z)$ and $\mathcal{I}, \beta \models \mathbf{trans}(\boldsymbol{Y}, \phi)$
  (by induction, since $\boldsymbol{Y} \notin \mathsf{FV}(\phi)$)
  $\Rightarrow \mathcal{I}, \beta_{[\boldsymbol{X} \mapsto \mathcal{U}']} \models \{\boldsymbol{X} \mid \forall r : \ \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, \phi)$
  (since $\boldsymbol{X} \notin \mathsf{FV}(\mathbf{trans}(\boldsymbol{Y}, \phi))$)
  $\Rightarrow \mathcal{I}, \beta_{[\boldsymbol{X} \mapsto \mathcal{U}']} \models \mathbf{trans}(\boldsymbol{X}, \forall r : \phi)$

- $\mathcal{U}' \subseteq [\![\psi \wedge \phi]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(\boldsymbol{X}) = \mathcal{U}'$
  $\Rightarrow \mathcal{U}' \subseteq [\![\psi]\!]^{\mathcal{I}}_{\alpha}$ and $\mathcal{U}' \subseteq [\![\phi]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \exists \beta.\ \mathcal{I}, \beta \models \mathbf{trans}(\boldsymbol{X}, \psi)$, $\beta(\boldsymbol{X}) = \mathcal{U}'$ and for each $z \in \mathsf{FV}(D)$,
  $\beta(z) = \alpha(z)$, and $\exists \beta'.\ \mathcal{I}, \beta' \models \mathbf{trans}(\boldsymbol{X}, \phi)$, $\beta'(\boldsymbol{X}) = \mathcal{U}'$ and
  for each $z \in \mathsf{FV}(D), \beta'(z) = \alpha(z)$
  (by induction)
  $\Rightarrow$ if $\alpha' = \beta_{[y_1 \mapsto \beta'(y_1), \ldots, y_n \mapsto \beta'(y_n), \boldsymbol{Z}_1 \mapsto \beta'(\boldsymbol{Z}_1), \ldots, \boldsymbol{Z}_n \mapsto \beta'(\boldsymbol{Z}_n)]}$, where
  $\{y_1, \ldots, y_n\} = \mathsf{FV}(\mathbf{trans}(\boldsymbol{X}, \phi)) \setminus \mathsf{FV}(D)$ and $\{z_1, \ldots, z_n\} =$
  $\mathsf{BV}(\mathbf{trans}(\boldsymbol{X}, \phi)) \setminus \{\boldsymbol{X}\}$, then $\mathcal{I}, \alpha' \models \mathbf{trans}(\boldsymbol{X}, \psi \wedge \phi)$
  (since $(\mathsf{FV}(\mathbf{trans}(\boldsymbol{X}, \psi)) \setminus \mathsf{FV}(D)) \cap (\mathsf{FV}(\mathbf{trans}(\boldsymbol{X}, \phi)) \setminus \mathsf{FV}(D))$
  $= \emptyset$ and for each $y \in \mathsf{FV}(D)$, $\beta(y) = \beta'(y)$)

Since $x \notin \mathsf{FV}(D_i)$, $u \in [\![D_i]\!]^{\mathcal{I}}_{\alpha_{[x \mapsto u]}}$, and thus, by the induction above, α' exists with $\alpha'(x) = u$ and $\mathcal{I}, \alpha' \models \mathbf{trans}(x, D_i)$.

□

Before we can define a normal form for sets of feature constraints, we will have to define some subsets of small variables and big variables:

Definition 4.11 (Main/Auxiliary Variables) *Let Σ be a set of feature constraints. If $x \mid y \in \Sigma$ and y occurs nowhere else in Σ, then y is an auxiliary variable in Σ. If z occurs in Σ and z is not an auxiliary variable, then it is a main variable in Σ. We will write "$z \in \mathsf{FV}(\Sigma)$" if z is a small variable that occurs in Σ and we will call the set of main variables of a feature clause Σ $\mathit{main}(\Sigma)$.*

Definition 4.12 (Constraint Variables) *The set* $\mathsf{CV}(\Sigma)$ *of constraint variables of a feature clause* Σ *is defined inductively as follows:*

- *If for some small variable* x, $\boldsymbol{X} - x \in \Sigma$, *then* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$.
- *If* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ *and for some* $f \in \mathcal{F}$, *for some big variable* $\boldsymbol{Y}$, $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma$, *then* $\boldsymbol{Y} \in \mathsf{CV}(\Sigma)$.

We will see that only those big variables which are constraint variables have to be considered in order to detect clashes in a feature clause.

In the following definition, it is not mentioned that a, b, t, t' have to be types, x, y, z, z' are small variables and $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{Z}$ are big variables, since this follows directly from the definition of feature constraints.

Definition 4.13 (Normal form) *A set of feature constraints* Σ *is said to be in normal form iff the following conditions hold:*

For each $f \in \mathcal{F}$, *for each* $r \in \mathcal{R}$,

1. (a) *If* x *is a main variable in* Σ, *then for some* $a \in \mathcal{T}$, $x \mid a \in \Sigma$.

   (b) *If* $\boldsymbol{X}$ *is a big variable in* Σ, *then for some* $a \in \mathcal{T}$, $\boldsymbol{X} \mid a \in \Sigma$.

2. (a) *If* $x \mid f : y$, $x \mid f : z \in \Sigma$, *then* $y = z$.

   (b) *If* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $\boldsymbol{X} \mid f : \boldsymbol{Y}$, $\boldsymbol{X} \mid f : \boldsymbol{Z} \in \Sigma$, *then* $\boldsymbol{Y} = \boldsymbol{Z}$.

3. (a) *If* $x \mid a$, $x \mid b \in \Sigma$, *then* $a = b$.

   (b) *If* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $\boldsymbol{X} \mid a$, $\boldsymbol{X} \mid b \in \Sigma$, *then* $a = b$.

4. (a) *If* $x \mid y \in \Sigma$ *and* $y \neq x$, *then* y *occurs nowhere else in* Σ.

   (b) *If* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ *and* $\boldsymbol{X} \mid y \in \Sigma$, *then* $\boldsymbol{X} - y \in \Sigma$.

   (c) *If* $\boldsymbol{X} - x \in \Sigma$, $\boldsymbol{X} \mid y \in \Sigma$ *and* $y \neq x$, *then* $y \mid x \in \Sigma$.

5. (a) *If* $x \mid \neg y \in \Sigma$, *then* $x \neq y$.

   (b) *If* $\boldsymbol{X} \mid \neg y \in \Sigma$, *then* $\boldsymbol{X} - y \notin \Sigma$.

6. (a) *If* $x \mid a$, $x \mid f : y \in \Sigma$, *then for each* $t, t' \in \mathcal{V}$ *s.t.* $t \preceq a$ *and* $t' \preceq a$, $\mathrm{approp}(t, f)\downarrow$, $\mathrm{approp}(t', f)\downarrow$ *and* $\mathrm{approp}(t, f) = \mathrm{approp}(t', f)$.

   (b) *If* $x \mid a$, $x \mid f : y$, $y \mid b \in \Sigma$, *then for each* $t \in \mathcal{V}$ *s.t.* $t \preceq a$, $\mathrm{approp}(t, f)\downarrow$ *and* $b \preceq \mathrm{approp}(t, f)$.

7. (a) *If* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $\boldsymbol{X} \mid a$, $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma$, *then for each* $t, t' \in \mathcal{V}$ *s.t.* $t \preceq a$ *and* $t' \preceq a$, $\mathrm{approp}(t, f)\downarrow$, $\mathrm{approp}(t', f)\downarrow$ *and* $\mathrm{approp}(t, f) = \mathrm{approp}(t', f)$.

(b) *If* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $\boldsymbol{X} \mid a$, $\boldsymbol{X} \mid f : \boldsymbol{Y}$, $\boldsymbol{Y} \mid b \in \Sigma$, *then for each* $t \in \mathcal{V}$ *s.t.* $t \preceq a$, $\mathit{approp}(t,f)\downarrow$ *and* $b \preceq \mathit{approp}(t,f)$.

8. (a) *If* $x \mid a$, $x \mid r : y \in \Sigma$, *then for each* $t, t' \in \mathcal{V}$ *s.t.* $t \preceq a$ *and* $t' \preceq a$, $\mathit{approp}(t,r)\downarrow$, $\mathit{approp}(t',r)\downarrow$ *and* $\mathit{approp}(t,r) = \mathit{approp}(t',r)$.

   (b) *If* $x \mid a$, $x \mid r : y$, $y \mid b \in \Sigma$, *then for each* $t \in \mathcal{V}$ *s.t.* $t \preceq a$, $\mathit{approp}(t,r)\downarrow$ *and* $b \preceq \mathit{approp}(t,r)$.

9. (a) *If* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $\boldsymbol{X} \mid a$, $\boldsymbol{X} \mid r : y \in \Sigma$, *then for each* $t, t' \in \mathcal{V}$ *s.t.* $t \preceq a$ *and* $t' \preceq a$, $\mathit{approp}(t,r)\downarrow$ *and* $\mathit{approp}(t',r)\downarrow$ *and* $\mathit{approp}(t,r) = \mathit{approp}(t',r)$.

   (b) *If* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $\boldsymbol{X} \mid a$, $\boldsymbol{X} \mid r : y$, $y \mid b \in \Sigma$, *then for each* $t \in \mathcal{V}$ *s.t.* $t \preceq a$, $\mathit{approp}(t,r)\downarrow$ *and* $b \preceq \mathit{approp}(t,r)$.

10. (a) *If* $x \mid \forall r : \boldsymbol{Y}, x \mid r : z \in \Sigma$, *then* $\boldsymbol{Y} - z \in \Sigma$.

    (b) *If* $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $\boldsymbol{X} \mid \forall r : \boldsymbol{Y}, \boldsymbol{X} \mid r : z \in \Sigma$, *then* $\boldsymbol{Y} - z \in \Sigma$.

11. *If* $\boldsymbol{X} \mid \forall r : \boldsymbol{Y}, z \mid r : z', \boldsymbol{X} - z \in \Sigma$, *then* $\boldsymbol{Y} - z' \in \Sigma$.

12. *If* $\boldsymbol{X} - y$, $\boldsymbol{X} \mid t \in \Sigma$, *then* $y \mid t' \in \Sigma$ *s.t.* $t' \preceq t$.

13. *If* $\boldsymbol{X} - y$, $\boldsymbol{X} \mid f : \boldsymbol{Z}$, $y \mid f : z' \in \Sigma$, *then* $\boldsymbol{Z} - z' \in \Sigma$.

14. *If* $\boldsymbol{X} - x, \boldsymbol{X} \mid r : z, x \mid \forall r : \boldsymbol{Y} \in \Sigma$, *then* $\boldsymbol{Y} - z \in \Sigma$.

Let's consider the properties of the normal form conditions. The conditions can be divided into conditions concerning feature constraints with small variables and conditions for feature constraints with constraint variables involved.

Conditions $1a$, $2a$, $3a$, $4a$, $5a$ and 6 deal with feature constraints with small variables only, as described in (Götz prep): The first condition ($1a$) ensures that each small variable is assigned a type. The second condition ($2a$) prevents that one feature has different (and possibly inconsistent) values, and by condition $3a$ we know that each small variable is associated with *one* type only. Feature constraints of the form $x \mid y$ only arise if a variable is translated by the function **trans**. Condition $4a$ states that a small variable does not occur more than once, i.e., that all other occurrences have been replaced. Inconsistencies like $x \mid \neg x$ are ruled out by condition $5a$. The sixth condition ensures that the appropriateness conditions are not violated.

$1b$, $2b$, $3b$, $4b$ and $4c$, $5b$ and 7 are the corresponding conditions for feature constraints with constraint variables. It should be clear that we only have to consider inconsistencies when we have feature constraints with constraint variables: Inconsistencies under the scope of a universal quantifier do not make a set of constraints inconsistent as long as there is *no* set-valued attribute which is concerned by the universal quantifier. If we have an (obviously) inconsistent feature term under the scope of universal quantification, say $\forall r : (\phi \wedge \neg\phi)$, then the only possibility for an inconsistency to arise is that $\forall r : (\phi \wedge \neg\phi)$ is part of an expression like $r : \psi \wedge \forall r : (\phi \wedge \neg\phi)$. $\forall r : (\phi \wedge \neg\phi)$ by itself is not inconsistent since we can construct a model in which no r–attributes occur. (If we consider the translation function **trans**, we see that big variables only arise if feature terms with a universal quantifier are translated.) If a big variable does *not* constrain any small variable (cf. definition 4.12), Σ cannot contain a clash. So, in our normal form system there is no condition which checks in *any* case the consistency of a constraint beginning with a big variable.

Conditions 8 and 9 correspond to 6 and 7 and ensure that the appropriateness conditions for set-valued attributes are not violated.

Conversely, condition 2 has no corresponding condition for set-valued features, neither for big nor for small variables. Since we can have several set-valued features r with different values, the implication in condition $2a$ or $2b$ clearly does not have to hold for set-valued attributes.

Conditions 10, 11, 12, 13 and 14 ensure that, if we have a feature constraint that constrains *all* set-valued attributes r and there *is* such an r, this information is propagated. Conditions 10, 11, 13 and 14 make big variables constraint variables, i.e., they simply check if there are constraints of the form $\boldsymbol{Y} - z$, and condition 12 is concerned with type constraints. $10a$ could be considered an initial case and $10b$ is the corresponding condition for constraint variables. Conditions 11, 13, 14 produce further constraints of the form $\boldsymbol{Y} - z$. Condition 12 could be seen as a “base case”. As we will see in the rewrite system, the trick with these conditions is that all clashes the corresponding rewrite rules potentially produce are detected by the other rules.

At first sight there might be missing a corresponding condition for condition 13 which, in the case where $\boldsymbol{X} - y$, $\boldsymbol{X} \mid r : \boldsymbol{Z} \in \Sigma$, introduces two further constraints of the form $y \mid r : z'$ and $\boldsymbol{Z} - z'$, provided we could have a feature constraint of the form $\boldsymbol{X} \mid r : \boldsymbol{Z}$, i.e., with a big variable instead of a small one. In our normal form we only consider the cases where clashes *can* occur. But since the satisfiability of a set of constraints is not touched in this case, I will come back to this point when I am defining an interpretation such that every feature clause in normal form is satisfied. With this intuition it becomes clear why – unlike other feature constraints containing big variables – a feature constraint beginning with a big variable and a set-valued attribute does not introduce further big variables. This can be seen in the translation function **trans**. Clearly, the same (that the satisfiability is not concerned) is true if $\boldsymbol{X} - y \in \Sigma$, $\boldsymbol{X} \mid f : \boldsymbol{Z} \in \Sigma$ and for no z', $y \mid f : z' \in \Sigma$.

We will now show that each feature clause, i.e., every set of feature constraints, in normal form is satisfiable by defining for each such clause an interpretation that satisfies it. In definition 4.14, we will define an interpretation which has exactly one object for each species in the signature. We will use this interpretation in order to build the interpretation in definition 4.16 which we will use to prove the satisfiability of a feature clause in normal form. For our purposes we will assume that the set of types is lexicographically ordered.

Definition 4.14 *For each signature* $S = \langle \mathcal{T}, \preceq, \mathcal{F}, \mathcal{R}, \mathrm{approp}\rangle$*, define an interpretation* $\mathcal{I}_S = \langle \mathcal{U}_S, \mathcal{S}_S, \mathcal{A}_S, \mathcal{D}_S\rangle$ *such that*

- $\mathcal{U}_S = \{u_t \mid t \in \mathcal{V}\}$
- $\mathcal{S}_S(u_t) = t$
- $\mathcal{A}_S(f)(u_t) = \begin{cases} u_{t'} & \text{if } \mathrm{approp}(t,f)\downarrow \text{ and } t' \text{ is the lex.} \\ & \text{smallest variety s.t. } t' \preceq \mathrm{approp}(t,f) \\ undefined & \text{otherwise} \end{cases}$
- $\mathcal{D}_S(r)(u_t) = \emptyset, \qquad$ *for all* $r \in \mathcal{R}$

Proposition 4.4 *For each signature* S*,* $\langle \mathcal{U}_S, \mathcal{S}_S, \mathcal{A}_S, \mathcal{D}_S\rangle$ *is an interpretation.*

Proof.

1. $\mathcal{U}_S$ is a set.
2. $\mathcal{S}_S$ is a total function from the set of objects to the set of varieties.
3. For each $u_t \in \mathcal{U}_S$,
   - $approp(\mathcal{S}_S(u_t), f)\downarrow$, $approp(\mathcal{S}_S(u_t), f) = t''$,
     $\Rightarrow \mathcal{A}_S(f)(u_t)\downarrow$, $\mathcal{A}_S(f)(u_t) = u_{t'}$ (by def. of $\mathcal{A}_S$)
     $\Rightarrow \mathcal{S}_S(\mathcal{A}_S(f)(u_t)) = t' \preceq t''$
   - $\mathcal{A}_S(f)(u_t)\downarrow$, $\mathcal{A}_S(f)(u_t) = u_{t'}$
     $\Rightarrow approp(\mathcal{S}_S(u_t), f)\downarrow$ (by def. of $\mathcal{A}_S$)
     $\Rightarrow \mathcal{S}(\mathcal{A}_S(f)(u_t)) = t' \preceq approp(\mathcal{S}_S(u_t), f)$
4. Since $\mathcal{D}_S(r)(u_t) = \emptyset$, both conditions on set-attribute interpretation functions hold.

□

Definition 4.15 *Let Σ be a feature clause in normal form over some signature $S = \langle \mathcal{T}, \preceq, \mathcal{F}, \mathcal{R}, \mathrm{approp} \rangle$. Let Σ' arise from Σ s.t.*

- *Each constraint of the form $x \mid a$ is replaced by $x \mid t$, where t is the lexicographically smallest variety s.t. $t \preceq a$.*
- *Each constraint of the form $\boldsymbol{X} \mid a$ s.t. for no $x \in \mathsf{FV}(\Sigma)$, $\boldsymbol{X} - x \in \Sigma$, is replaced by $\boldsymbol{X} \mid t$, where t is the lexicographically smallest variety s.t. $t \preceq a$.*

Clearly, if Σ is in normal form, then Σ' is in normal form as well. By transitivity of $\preceq$, conditions 6, 7, 8 and 9 still hold. Conditions $3a$ and $3b$ cannot be violated, since the constraints are replaced and not added. Condition 12 will still hold, since we only change a constraint $\boldsymbol{X} \mid a$, if $\boldsymbol{X}$ does *not* constrain a small variable y. None of the other conditions are concerned with constraints of the form $x \mid a$ and $\boldsymbol{X} \mid a$.

Definition 4.16 *Let Σ be a feature clause in normal form and Σ' constructed from Σ as in definition 4.15. Then define an interpretation $\mathcal{I}_T = \langle \mathcal{U}_T, \mathcal{S}_T, \mathcal{A}_T, \mathcal{D}_T \rangle$ such that*

- $\mathcal{U}_T = \mathcal{U}_S \cup \{x \mid x \in \mathsf{FV}(\Sigma') \textit{ and for some } a, x \mid a \in \Sigma'\} \cup$
  $\{u_{\mathbf{x}} \mid \boldsymbol{X} \in \mathsf{CV}(\Sigma') \textit{ for no } x \in \mathsf{FV}(\Sigma),\ \boldsymbol{X} - x \in \Sigma,$
  $\textit{and for some } a,\ \boldsymbol{X} \mid a \in \Sigma'\}$
- $\mathcal{S}_T(u) = \begin{cases} S_S(u) & \textit{if } u \in \mathcal{U}_S \\ t & \textit{if } u = x \textit{ and } x \mid t \in \Sigma' \\ t & \textit{if } u = u_{\mathbf{x}} \textit{ and } \boldsymbol{X} \mid t \in \Sigma' \end{cases}$

$$\bullet\ \mathcal{A}_T(f)(u) = \begin{cases} \mathcal{A}_S(f)(u) & \textit{if } u \in \mathcal{U}_S \textit{ and } \mathcal{A}_S(f)(u)\downarrow \\ y & \textit{if } u = x \textit{ and } x \mid f : y \in \Sigma' \\ u_{t'} & \textit{if } u = x \textit{ for some } x \in \mathsf{FV}(\Sigma')\ , \\ & x \mid t \in \Sigma', \textit{ for no } y \textit{ is } x \mid f : y \in \Sigma', \\ & \ \mathrm{approp}(t,f)\downarrow \textit{ and } t' \in \mathcal{V} \textit{ is the lex.} \\ & \ \textit{smallest variety s.t. } t' \preceq \mathrm{approp}(t,f) \\ & \textit{and there is no } \boldsymbol{Y} \textit{ s.t. for some } \boldsymbol{X}, \\ & \boldsymbol{X} - x \in \Sigma' \text{ and } \boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma' \\ u_{\mathbf{y}} & \textit{if } u = x, \boldsymbol{X} - x \in \Sigma',\ \boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma' \\ & \textit{for no } w,\ x \mid f : w \in \Sigma', \\ & \textit{for no } y,\ \boldsymbol{Y} \mid y \in \Sigma', \\ & \textit{for no } v,\ \boldsymbol{Y} - v \in \Sigma' \\ v & \textit{if } u = x, \boldsymbol{X} - x \in \Sigma',\ \boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma', \\ & \textit{for no } w,\ x \mid f : w \in \Sigma', \\ & \textit{for no } y,\ \boldsymbol{Y} \mid y \in \Sigma', \\ & \boldsymbol{Y} - v \in \Sigma' \\ & \textit{and for each } v',\ \textit{if } \boldsymbol{Y} - v' \in \Sigma', \\ & \textit{then } v \textit{ is lex. smaller than } v' \\ u_{\mathbf{y}} & \textit{if } u = u_{\mathbf{x}},\ \boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma', \\ & \textit{for no } y,\ \boldsymbol{Y} \mid y \in \Sigma' \\ & \textit{for no } v,\ \boldsymbol{Y} - v \in \Sigma' \\ v & \textit{if } u = u_{\mathbf{x}},\ \boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma', \\ & \textit{for no } y,\ \boldsymbol{Y} \mid y \in \Sigma', \\ & \boldsymbol{Y} - v \in \Sigma', \\ & \textit{and for each } v',\ \textit{if } \boldsymbol{Y} - v' \in \Sigma', \\ & \textit{then } v \textit{ is lex. smaller than } v' \\ y & \textit{if } u = x,\ \boldsymbol{X} - x \in \Sigma', \boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma', \\ & \boldsymbol{Y} \mid y \in \Sigma',\ \textit{and for no } w, \\ & x \mid f : w \in \Sigma' \\ y & \textit{if } u = u_{\mathbf{x}},\ \boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'\ , \\ & \ \boldsymbol{Y} \mid y \in \Sigma' \\ undefined & otherwise \end{cases}$$

- *If* $u = x$ *then* $\mathcal{D}_T(r)(u)$ *is the smallest set s.t.*
  - *if* $x \mid r : y \in \Sigma'$, *then* $y \in \mathcal{D}_T(r)(x)$
  - *if* $\boldsymbol{X} - x \in \Sigma', \boldsymbol{X} \mid r : z \in \Sigma'$, *then* $z \in \mathcal{D}_T(r)(x)$[10]

  If $u = u_{\mathbf{x}}$ *then* $\mathcal{D}_T(r)(u)$ *is the smallest set s.t.*

  - *if* $\boldsymbol{X} \mid r : z \in \Sigma'$, *then* $z \in \mathcal{D}_T(r)(u_{\mathbf{x}})$

  If $u \neq x$ *and* $u \neq u_{\mathbf{x}}$ *then* $\mathcal{D}_T(r)(u) = \emptyset$.

Proposition 4.5 *For each set of constraints* Σ, *if* Σ *is in normal form and* Σ' *is defined as above, then* $\mathcal{I}_T = \langle \mathcal{U}_T, \mathcal{S}_T, \mathcal{A}_T, \mathcal{D}_T \rangle$ *is an interpretation.*

Proof.

1. $\mathcal{U}_T$ is a set.
2. $\mathcal{S}_T$ is a total function from the set of objects to the set of varieties.
3. For each $u \in \mathcal{U}_T$,
   (a) If $u \in \mathcal{U}_S$ then
   - *approp*$(\mathcal{S}_T(u), f) \downarrow$, *approp*$(\mathcal{S}_T(u), f) = t$,
     $\Rightarrow \mathcal{A}_T(f)(u) \downarrow$ (by def. of $\mathcal{A}_T$)
     $\Rightarrow \mathcal{S}_T(\mathcal{A}_T(f)(u)) \preceq t$
   - $\mathcal{A}_T(f)(u) \downarrow$
     $\Rightarrow$ *approp*$(\mathcal{S}_T(u), f) \downarrow$ (by def. of $\mathcal{A}_T$)
     $\Rightarrow \mathcal{S}(\mathcal{A}_T(f)(u)) \preceq$ *approp*$(\mathcal{S}_T(u), f)$

   (b) If $u = x$ then

   i. If for some y, $x \mid f : y \in \Sigma'$, then
   - *approp*$(\mathcal{S}_T(x), f) \downarrow$, *approp*$(\mathcal{S}_T(x), f) = t''$
     $\Rightarrow \mathcal{A}_T(f)(x) \downarrow$ (by def. of $\mathcal{A}_T$)
     $\Rightarrow$ for some t', $y \mid t' \in \Sigma'$ (by cond. 1)
     $\Rightarrow \mathcal{S}_T(\mathcal{A}_T(f)(x)) = \mathcal{S}_T(y) = t'$
     $\Rightarrow \mathcal{S}_T(\mathcal{A}_T(f)(x)) \preceq t''$ (by cond. 6b)
   - $\mathcal{A}_T(f)(x) \downarrow$
     $\Rightarrow \mathcal{A}_T(f)(x) = y$ (by def. of $\mathcal{A}_T$ and cond. 2a)
     $\Rightarrow$ for some t',$x \mid t' \in \Sigma'$ (by cond. 1)
     $\Rightarrow$ for some t'',$y \mid t'' \in \Sigma'$ (by cond. 1)
     $\Rightarrow$ *approp*$(\mathcal{S}_T(x), f) \downarrow$ (by cond. 6a)

[10] Hence, if $x \mid \forall r : \boldsymbol{Y} \in \Sigma'$ but there is no z, s.t. $x \mid r : z \in \Sigma'$, then $\boldsymbol{Y} - z \notin \Sigma$ and $\mathcal{D}(r)(x) = \emptyset$.

$\Rightarrow \mathcal{S}_T(\mathcal{A}_T(f)(x)) = \mathcal{S}_T(y) = t''$
(by def. of $\mathcal{S}_T$ and cond. $3a$)
$\Rightarrow \mathcal{S}_T(y) \preceq approp(\mathcal{S}_T(x), f)$ (by cond. $6b$)

ii. If $x \mid t \in \Sigma'$, for no $\boldsymbol{X}$, for no $\boldsymbol{Y}$, $\boldsymbol{X} - x \in \Sigma'$ *and* $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'$, and there is no $\boldsymbol{Y}$ s.t. for some $\boldsymbol{X}$, $\boldsymbol{X} - x \in \Sigma'$ *and* $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'$ then

- $approp(\mathcal{S}_T(x), f)\downarrow$, $approp(\mathcal{S}_T(x), f) = t''$
  $\Rightarrow \mathcal{A}_T(f)(x)\downarrow$
  $\Rightarrow \mathcal{A}_T(f)(x) = u_{t'}$ (by def. of $\mathcal{A}_T$)
  $\Rightarrow \mathcal{S}_T(\mathcal{A}_T(f)(x)) = t' \preceq approp(t, f)$
- $\mathcal{A}_T(f)(x)\downarrow$
  $\Rightarrow \mathcal{A}_T(f)(x) = u_{t'}$ (by def. of $\mathcal{A}_T$)
  $\Rightarrow \mathcal{S}_T(\mathcal{A}_T(f)(x)) = t'$ (by def. of $\mathcal{S}_T$)
  $\Rightarrow approp(t, f)\downarrow$ (by def. of $\mathcal{A}_T$)
  $\Rightarrow t' \preceq approp(\mathcal{S}_T(x), f)$

iii. If $\boldsymbol{X} - x \in \Sigma'$, $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'$, there is no y, s.t. $\boldsymbol{Y} \mid y \in \Sigma'$, there is no w s.t. $x \mid f : w \in \Sigma'$, and there is no v s.t. $\boldsymbol{Y} - v \in \Sigma$, then

- $approp(\mathcal{S}_T(x), f)\downarrow$
  $\Rightarrow$ for some t, $x \mid t \in \Sigma'$
  $\Rightarrow$ for some t', $approp(t, f) = t'$
  $\Rightarrow$ for some t'', $\boldsymbol{Y} \mid t'' \in \Sigma'$ (by cond. 1)
  $\Rightarrow \mathcal{A}_T(f)(x)\downarrow$
  $\Rightarrow \mathcal{A}_T(f)(x) = u_{\mathbf{y}}$ (by def. of $\mathcal{A}_T$)
  $\Rightarrow \mathcal{S}_T(u_{\mathbf{y}}) = t''$ (by def. of $\mathcal{S}_T$ and cond. $3b$)
  $\Rightarrow t'' \preceq t'$ (by conds. $7a$ and $7b$)
- $\mathcal{A}(f)(x)\downarrow$
  $\Rightarrow$ for some t', t'', $\boldsymbol{X} \mid t'$, $\boldsymbol{Y} \mid t'' \in \Sigma'$ (by cond. $1b$)
  $\Rightarrow \mathcal{A}_T(f)(x) = u_{\mathbf{y}}$
  (by def. of $\mathcal{A}_T$ and cond. $3b$)
  $\Rightarrow approp(\mathcal{S}_T(t'), f)\downarrow$ (by cond. $7a$)
  $\Rightarrow \mathcal{S}_T(x) \preceq t'$ (by def. of $\mathcal{S}_T$ and cond. 12)
  $\Rightarrow approp(\mathcal{S}_T(x), f)\downarrow$
  $\Rightarrow \mathcal{S}_T(\mathcal{A}_T(f)(x)) = t'' \preceq approp(\mathcal{S}_T(x), f)$
  (by cond. $7b$)

iv. If $\boldsymbol{X} - x \in \Sigma'$, $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'$, there is no y, s.t. $\boldsymbol{Y} \mid y \in \Sigma'$, there is no w s.t. $x \mid f : w \in \Sigma'$, $\boldsymbol{Y} - v \in \Sigma'$ and for each v', if $\boldsymbol{Y} - v' \in \Sigma'$, then v is lex. smaller than v',
then

- $approp(\mathcal{S}_T(x), f)\downarrow$
  $\Rightarrow$ for some t,t'',t''',$\bar{t}$, $x \mid t, v \mid t''$, $\boldsymbol{Y} \mid t'''$, $\boldsymbol{X} \mid \bar{t} \in \Sigma'$

(by cond. 1)

$\Rightarrow$ for some t', $approp(t, f) = t'$

$\Rightarrow \mathcal{A}_T(f)(x) \downarrow$

$\Rightarrow \mathcal{A}_T(f)(x) = v$ (by def. of $\mathcal{A}_T$)

$\Rightarrow \mathcal{S}_T(v) = t''$ (by def. of $\mathcal{S}_T$)

$\Rightarrow t'' \preceq t'''$ (by cond. 12)

$\Rightarrow t \preceq \bar{t}$ (by cond. 12)

$\Rightarrow approp(t, f) = approp(\bar{t}, f) = t'$ (by cond. $7a$)

$\Rightarrow t''' \preceq t'$ (by cond. $7b$)

$\Rightarrow t'' \preceq t'$

- $\mathcal{A}(f)(x) \downarrow$

  $\Rightarrow$ for some $\bar{t}, t''$, $\boldsymbol{Y} \mid t''$, $\boldsymbol{X} \mid \bar{t} \in \Sigma'$ (by cond. $1b$)

  $\Rightarrow \mathcal{A}_T(f)(x) = v$

  (by def. of $\mathcal{A}_T$)

  $\Rightarrow approp(\bar{t}, f) \downarrow$ (by cond. $7a$)

  $\Rightarrow \mathcal{S}_T(x) \preceq \bar{t}$ (by def. of $\mathcal{S}_T$ and cond. 12)

  $\Rightarrow approp(\mathcal{S}_T(x), f) = approp(\bar{t}, f)$ (by cond. $7a$)

  $\Rightarrow \mathcal{S}_T(\mathcal{A}_T(f)(x)) = t'' \preceq approp(\mathcal{S}_T(x), f)$

  (by cond. $7b$)

v. If $\boldsymbol{X} - x \in \Sigma', \boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'$, $\boldsymbol{Y} \mid y \in \Sigma'$ and there is no w s.t $x \mid f : w \in \Sigma'$ then

- $approp(\mathcal{S}_T(x), f) \downarrow$

  $\Rightarrow$ for some t, t', t'', $x \mid t, y \mid t'$, $\boldsymbol{Y} \mid t'' \in \Sigma'$

  (by conds. $1a$, $1b$)

  $\Rightarrow$ for some t''', $approp(t, f) = t'''$

  $\Rightarrow \mathcal{A}_T(f)(x) \downarrow$

  $\Rightarrow \mathcal{A}_T(f)(x) = y$ (by def. of $\mathcal{A}_T$)

  $\Rightarrow \mathcal{S}_T(y) = t'$

  $\Rightarrow \boldsymbol{Y} - y \in \Sigma'$ (by cond. $4b$)

  $\Rightarrow t' \preceq t''$ (by cond. 12)

  $\Rightarrow t'' \preceq t'''$ (by cond. $7b$)

  $\Rightarrow t' \preceq t'''$

- $\mathcal{A}(f)(x) \downarrow$

  $\Rightarrow \mathcal{A}_T(f)(x) = y$

  (by def. of $\mathcal{A}_T$ and conds. $4a$, $4b$, $4c$)

  $\Rightarrow$ for some $t, t', t'', \boldsymbol{X} \mid t, y \mid t'$, $\boldsymbol{Y} \mid t'' \in \Sigma'$

  (by conds. $1a$, $1b$)

  $\Rightarrow \boldsymbol{Y} - y \in \Sigma'$ (by cond. $4b$)

  $\Rightarrow t' \preceq t''$ (by cond. 12)

  $\Rightarrow \mathcal{S}_T(x) \preceq t$ (by cond. 12)

  $\Rightarrow approp(\mathcal{S}_T(x), f) \downarrow$ (by conds. $6a$, 12)

  $\Rightarrow t'' \preceq approp(\mathcal{S}_T(x), f)$ (by cond. $7b$)

$\Rightarrow t' \preceq approp(\mathcal{S}_T(x), f)$

(c) i. If $u = u_{\mathbf{x}}$, $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'$, for no y, $\boldsymbol{Y} \mid y \in \Sigma'$ and there is no v s.t. $\boldsymbol{Y} - v \in \Sigma$,
then

- $approp(\mathcal{S}_T(u_{\mathbf{x}}), f)\downarrow$, $approp(\mathcal{S}_T(u_{\mathbf{x}}), f) = t$
$\Rightarrow \mathcal{S}_T(u_{\mathbf{x}}) = t'$, $\boldsymbol{X} \mid t' \in \Sigma'$ (by cond. 1b)
$\Rightarrow \mathcal{A}_T(f)(u_{\mathbf{x}})\downarrow$
$\Rightarrow \mathcal{A}_T(f)(u_{\mathbf{x}}) = u_{\mathbf{y}}$ (by def. of $\mathcal{A}_T$)
$\Rightarrow S_T(\mathcal{A}_T(f)(u_{\mathbf{x}})) = t''$ s.t. $\boldsymbol{Y} \mid t'' \in \Sigma'$
$\Rightarrow t'' \preceq t$ (by cond. 7b)
- $\mathcal{A}_T(f)(u_{\mathbf{x}})\downarrow$
$\Rightarrow \mathcal{A}_T(f)(u_{\mathbf{x}}) = u_{\mathbf{y}}$ (by def. of $\mathcal{A}_T$)
$\Rightarrow approp(\mathcal{S}_T(u_{\mathbf{x}}), f)\downarrow$ (by cond. 7a)
$\Rightarrow S_T(\mathcal{A}_T(f)(u_{\mathbf{x}})) = t''$ s.t. $\boldsymbol{Y} \mid t'' \in \Sigma'$
$\Rightarrow t'' \preceq approp(\mathcal{S}_T(u_{\mathbf{x}}), f)$ (by cond. 7b)

ii. If $u = u_{\mathbf{x}}$, $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'$, for no y, $\boldsymbol{Y} \mid y \in \Sigma'$, $\boldsymbol{Y} - v \in \Sigma'$ and for each v', if $\boldsymbol{Y} - v' \in \Sigma'$, then v is lex. smaller than v',
then

- $approp(\mathcal{S}_T(u_{\mathbf{x}}), f)\downarrow$, $approp(\mathcal{S}_T(u_{\mathbf{x}}), f) = t$
$\Rightarrow$ for some t', t''', $\boldsymbol{X} \mid t'$, $\boldsymbol{Y} \mid t''' \in \Sigma'$ (by cond. 1b)
$\Rightarrow \mathcal{S}_T(u_{\mathbf{x}}) = t'$ (by def. of $\mathcal{S}_T$)
$\Rightarrow \mathcal{A}_T(f)(u_{\mathbf{x}})\downarrow$
$\Rightarrow \mathcal{A}_T(f)(u_{\mathbf{x}}) = v$ (by def. of $\mathcal{A}_T$)
$\Rightarrow S_T(\mathcal{A}_T(f)(u_{\mathbf{x}})) = t''$ s.t. $v \mid t'' \in \Sigma'$
$\Rightarrow t'' \preceq t'''$ (by cond. 12)
$\Rightarrow t''' \preceq t$ (by cond. 7b)
$\Rightarrow t'' \preceq t$
- $\mathcal{A}_T(f)(u_{\mathbf{x}})\downarrow$
$\Rightarrow$ for some t, t', t'', $\boldsymbol{X} \mid t, v \mid t'$, $\boldsymbol{Y} \mid t'' \in \Sigma'$
(by conds. 1a, 1b)
$\Rightarrow \mathcal{A}_T(f)(u_{\mathbf{x}}) = v$ (by def. of $\mathcal{A}_T$)
$\Rightarrow approp(t, f)\downarrow$ (by cond. 7a)
$\Rightarrow \mathcal{S}_T(v) = t'$ (by def. of $\mathcal{S}_T$)
$\Rightarrow t' \preceq t''$ (by cond. 12)
$\Rightarrow t'' \preceq approp(t, f)$ (by cond. 7b)
$\Rightarrow t' \preceq approp(t, f)$

iii. If $u = u_{\mathbf{x}}$, $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'$, $\boldsymbol{Y} \mid y \in \Sigma'$ then

- $approp(\mathcal{S}_T(u_{\mathbf{x}}), f)\downarrow$, $approp(\mathcal{S}_T(u_{\mathbf{x}}), f) = t$
$\Rightarrow \mathcal{S}_T(u_{\mathbf{x}}) = t'$, s.t. $\boldsymbol{X} \mid t' \in \Sigma'$ (by cond. 1b)
$\Rightarrow \mathcal{A}_T(f)(u_{\mathbf{x}})\downarrow$
$\Rightarrow \mathcal{A}_T(f)(u_{\mathbf{x}}) = y$ (by def. of $\mathcal{A}_T$)

$\Rightarrow S_T(\mathcal{A}_T(f)(u_{\mathbf{x}})) = t''$ s.t. $y \mid t'' \in \Sigma'$
$\Rightarrow \boldsymbol{Y} - y \in \Sigma'$ (by cond. $4b$)
$\Rightarrow t'' \preceq t'''$ s.t. $\boldsymbol{Y} \mid t''' \in \Sigma'$ (by cond. 12)
$\Rightarrow t''' \preceq t$ (by cond. $7b$)
$\Rightarrow t'' \preceq t$

- $\mathcal{A}_T(f)(u_{\mathbf{x}}) \downarrow$
$\Rightarrow \mathcal{A}_T(f)(u_{\mathbf{x}}) = y$
$\Rightarrow \mathit{approp}(\mathcal{S}_T(u_{\mathbf{x}}), f) \downarrow$ (by cond. $7a$)
$\Rightarrow S_T(\mathcal{A}_T(f)(u_{\mathbf{x}})) = t''$ s.t. $y \mid t'' \in \Sigma'$
$\Rightarrow t'' \preceq \mathit{approp}(\mathcal{S}_T(u_{\mathbf{x}}), f)$ (by conds. $7b$ and $4b$)

Thus, $\mathcal{A}_T(f)(u)$ is an attribute interpretation function.

4. For each $u \in \mathcal{U}_T$,

(a) If $u \neq x$ and $u \neq u_{\mathbf{x}}$ then

- $\mathit{approp}(\mathcal{S}_T(u), r) \downarrow$ and $\mathit{approp}(\mathcal{S}_T(u), r) = t$
$\Rightarrow$ for each $u' \in \mathcal{D}_T(r)(u)$, $\mathcal{S}_T(u') \preceq t$
(since $\mathcal{D}_T(r)(u) = \emptyset$)

(b) If $u = x$ then

- $\mathit{approp}(\mathcal{S}_T(x), r) \downarrow$ and $\mathit{approp}(\mathcal{S}_T(x), r) = t$
$\Rightarrow$ for each $y \in \mathcal{D}_T(r)(x)$, $\mathcal{S}_T(y) \preceq t$
(by conds. $8a$ and $8b$)
- $\mathcal{D}_T(r)(x) \neq \emptyset$
$\Rightarrow$ for each $u \in \mathcal{D}_T(r)(x)$, $u = y$, s.t. $x \mid r : y \in \Sigma'$ or
$u = z$, s.t. $\boldsymbol{X} - x, \boldsymbol{X} \mid r : z \in \Sigma'$
$\Rightarrow$ for some t, $\boldsymbol{X} \mid t \in \Sigma'$ (by cond. $1b$)
$\Rightarrow \mathit{approp}(\mathcal{S}_T(x), r) \downarrow$ and $\mathcal{S}_T(y) \preceq \mathit{approp}(\mathcal{S}_T(x), r)$
(by cond. $8b$)
$\Rightarrow \mathit{approp}(t, r) \downarrow$ (by cond. $9b$)
$\Rightarrow \mathcal{S}_T(x) = t$ (by cond. 12 and construction of Σ')
$\Rightarrow \mathcal{S}_T(z) \preceq \mathit{approp}(t, r)$ (by cond. $9b$)
$\Rightarrow \mathcal{S}_T(z) \preceq \mathit{approp}(\mathcal{S}_T(x), r)$ (by cond. $8a$)

(c) If $u = u_{\mathbf{x}}$ then

- $\mathit{approp}(\mathcal{S}_T(u_{\mathbf{x}}), r) \downarrow$, $\mathit{approp}(\mathcal{S}_T(u_{\mathbf{x}}), r) = t$
$\Rightarrow$ for each $z \in \mathcal{D}_T(r)(u_{\mathbf{x}})$, $\mathcal{S}_T(z) \preceq t$
(by conds. $9a$ and $9b$, since $\boldsymbol{X} \in \mathsf{CV}(\Sigma')$)
- $\mathcal{D}_T(r)(u_{\mathbf{x}}) \neq \emptyset$
$\Rightarrow$ for each $u \in \mathcal{D}_T(r)(u_{\mathbf{x}}), \mathcal{S}_T(u) \preceq \mathit{approp}(\mathcal{S}_T(u_{\mathbf{x}}), r)$
(by cond. $9b$)

Thus, $\mathcal{D}_T(r)(u)$ is a set-attribute interpretation function.

□

Proposition 4.6 *For each feature clause Σ, if Σ is in normal form, then Σ is satisfiable.*

Proof. We prove that Σ', which is constructed from a set of constraints Σ in normal form as in definition 4.15, is satisfiable, since the satisfiability of Σ follows from the satisfiability of Σ'.[11]

1. By proposition 4.5, $\mathcal{I}_T$ is an interpretation.

2. Let β be some variable assignment. Define
$$\alpha(x) = \begin{cases} x & \text{if } x \text{ is a main variable in } \Sigma' \\ y & \text{if } x \text{ is an auxiliary variable and } y \mid x \in \Sigma' \\ \beta(x) & \text{otherwise} \end{cases}$$
Define
$$\alpha(\boldsymbol{X}) = \begin{cases} \{y \mid \boldsymbol{X} - y \in \Sigma'\} \cup \{u_{\boldsymbol{X}}\} & \text{if } \boldsymbol{X} \in \mathsf{CV}(\Sigma') \text{ and} \\ & \text{for no } y,\ \boldsymbol{X} \mid y \in \Sigma' \\ \{y \mid \boldsymbol{X} - y \in \Sigma'\} & \text{otherwise} \end{cases}$$

3. Now we have to show that for each $\sigma \in \Sigma'$, $\mathcal{I}, \alpha \models \sigma$ and hence, $\mathcal{I}, \alpha \models \Sigma$:

   - $x \mid y \in \Sigma'$
     $\Rightarrow \alpha(x) = x,\ \alpha(y) = x$ (by definition of α)
   - $x \mid \neg y \in \Sigma'$
     $\Rightarrow \alpha(x) = x,\ \alpha(y) = y$ (by definition of α)
   - $x \mid t \in \Sigma'$
     $\Rightarrow \mathcal{S}(\alpha(x)) = t \preceq t$
   - $x \mid f : y \in \Sigma'$
     $\Rightarrow \alpha(x) = x,\ \alpha(y) = y$
     $\Rightarrow \mathcal{A}(f)(x) = y$
   - $x \mid r : y \in \Sigma'$
     $\Rightarrow \alpha(x) = x, \alpha(y) = y$
     $\Rightarrow y \in \mathcal{D}(r)(x)$
   - $x \mid \forall r : \boldsymbol{Y} \in \Sigma'$
     $\Rightarrow \alpha(x) = x$
     $\Rightarrow \alpha(\boldsymbol{Y}) = \{y \mid \boldsymbol{Y} - y \in \Sigma'\}$
     or $\alpha(\boldsymbol{Y}) = \{y \mid \boldsymbol{Y} - y \in \Sigma'\} \cup \{u_{\mathbf{y}}\}$
     - $x \mid r : y \in \Sigma'$
       $\Rightarrow \boldsymbol{Y} - y \in \Sigma'$ (by cond. $10a$)
       $\Rightarrow y \in \mathcal{D}(r)(x)$

[11] For each interpretation, for each variable assignment, if $\mathcal{I}, \alpha \models x \mid t$ and $t \preceq a$ then $\mathcal{I}, \alpha \models x \mid a$, and if $\mathcal{I}, \alpha \models \boldsymbol{X} \mid t$ and $t \preceq a$ then $\mathcal{I}, \alpha \models \boldsymbol{X} \mid a$.

– for some $\boldsymbol{X}$, $\boldsymbol{X} - x$ and $\boldsymbol{X} \mid r : z \in \Sigma'$
$\Rightarrow \boldsymbol{Y} - z \in \Sigma'$ (by cond. 14)
$\Rightarrow z \in \mathcal{D}(r)(x)$

$\Rightarrow \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(x)$

- $\boldsymbol{X} - y \in \Sigma'$
$\Rightarrow \alpha(y) = y$
$\Rightarrow y \in \alpha(\boldsymbol{X})$ (by definition of α)
- $\boldsymbol{X} \mid y \in \Sigma'$
$\Rightarrow \alpha(y) = y$
$\Rightarrow \alpha(\boldsymbol{X}) = \{y\}$ or $\alpha(\boldsymbol{X}) = \emptyset$
(by conds. $4a$, $4b$, $4c$ and definition of α)
- $\boldsymbol{X} \mid \neg y \in \Sigma'$
$\Rightarrow \alpha(y) = y$
$\Rightarrow y \notin \alpha(\boldsymbol{X})$ (by cond. $5b$ and definition of α)
- $\boldsymbol{X} \mid t \in \Sigma'$
$\Rightarrow \alpha(\boldsymbol{X}) = \{y \mid \boldsymbol{X} - y \in \Sigma'\} \cup \{u_{\mathbf{x}}\}$
or $\alpha(\boldsymbol{X}) = \{y \mid \boldsymbol{X} - y \in \Sigma'\}$
$\Rightarrow \mathcal{S}(u_{\mathbf{x}}) = t$, $\mathcal{S}(y) \preceq t$
(by cond. 12 and definition of $\mathcal{S}$)
- $\boldsymbol{X} \mid f : \boldsymbol{Y} \in \Sigma'$
We have to show that for each $u \in \alpha(\boldsymbol{X})$, $\mathcal{A}(f)(u) \downarrow$ and
$\alpha(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$.
  – First, we need to ensure that
  for each $u \in \alpha(\boldsymbol{X})$, $\mathcal{A}(f)(u) \downarrow$ and $u \in \alpha(\boldsymbol{Y})$:

  $\alpha(\boldsymbol{X}) = \{x \mid \boldsymbol{X} - x \in \Sigma'\} \cup \{u_{\mathbf{x}}\}$
  (if $\boldsymbol{X} \in \mathsf{CV}(\Sigma')$ and for no x, $\boldsymbol{X} \mid x \in \Sigma'$)

  or

  $\alpha(\boldsymbol{X}) = \{x \mid \boldsymbol{X} - x \in \Sigma'\}$

  (if $\boldsymbol{X} \notin \mathsf{CV}(\Sigma')$ or for some x, $\boldsymbol{X} \mid x \in \Sigma'$)
  $\Rightarrow$ for each u $\in \alpha(\boldsymbol{X})$
    * $u = u_{\mathbf{x}}$
    $\Rightarrow \mathcal{A}(f)(u_{\mathbf{x}}) \downarrow$ and
      · $\mathcal{A}(f)(u_{\mathbf{x}}) = u_{\mathbf{y}}$ (by def. of $\mathcal{A}$)
      $\Rightarrow u_{\mathbf{y}} \in \alpha(\boldsymbol{Y})$ (since $\boldsymbol{X} \in \mathsf{CV}(\Sigma') \rightarrow \boldsymbol{Y} \in \mathsf{CV}(\Sigma')$
      and, by def. of $\mathcal{A}$, there is no y s.t. $\boldsymbol{Y} - y \in \Sigma'$)
      · or $\mathcal{A}(f)(u_{\mathbf{x}}) = v$
      $\Rightarrow \boldsymbol{Y} - v \in \Sigma'$
      $\Rightarrow v \in \alpha(\boldsymbol{Y})$ (by def. of α)

* $u = x$ s.t. $\boldsymbol{X} - x \in \Sigma'$
  $\Rightarrow \mathcal{A}(f)(x)\downarrow$ and
  · $\mathcal{A}(f)(u) = y$ s.t. $x \mid f : y \in \Sigma'$
    $\Rightarrow \boldsymbol{Y} - y \in \Sigma'$ (by cond. 13)
    $\Rightarrow y \in \alpha(\boldsymbol{Y})$
  · or $\mathcal{A}(f)(u) = y$
    $\Rightarrow \boldsymbol{Y} \mid y \in \Sigma'$ (by def. of $\mathcal{A}$)
    $\Rightarrow \boldsymbol{Y} - y \in \Sigma'$ (by cond. $4b$)
    $\Rightarrow y \in \alpha(\boldsymbol{Y})$
  · or $\mathcal{A}(f)(u) = u_{\boldsymbol{Y}}$
    $\Rightarrow$ there is no y s.t. $\boldsymbol{Y} \mid y \in \Sigma'$ (by def. of $\mathcal{A}$)
    $\Rightarrow u_{\mathbf{y}} \in \alpha(\boldsymbol{Y})$ (since $\boldsymbol{X} \in \mathsf{CV}(\Sigma') \to \boldsymbol{Y} \in \mathsf{CV}(\Sigma')$
    and, by def. of $\mathcal{A}$, there is no y s.t. $\boldsymbol{Y} - y \in \Sigma'$)

– Secondly, we have to show that for each $u' \in \alpha(\boldsymbol{Y})$, there is a $u \in \alpha(\boldsymbol{X})$ s.t. $\mathcal{A}(f)(u) = u'$:

  $\alpha(\boldsymbol{Y}) = \{y \mid \boldsymbol{Y} \mid y \in \Sigma'\} \cup \{u_{\mathbf{y}}\}$
  (if $\boldsymbol{Y} \in \mathsf{CV}(\Sigma')$ and for no y, $\boldsymbol{Y} \mid y \in \Sigma'$)

  or $\alpha(\boldsymbol{Y}) = \{y \mid \boldsymbol{Y} - y \in \Sigma'\}$
  (if $\boldsymbol{Y} \notin \mathsf{CV}(\Sigma')$ or for some y, $\boldsymbol{Y} - y \in \Sigma'$)
  $\Rightarrow$ for each $u' \in \alpha(\boldsymbol{Y})$

  * $u' = u_{\mathbf{y}}$

    $\Rightarrow \exists u.\ u = x$ and $\boldsymbol{X} - x \in \Sigma'$, there is no w s.t. $x \mid f : w \in \Sigma'$, and there is no y s.t. $\boldsymbol{Y} \mid y \in \Sigma'$
    $\Rightarrow \mathcal{A}(f)(x) = u_{\mathbf{y}}$
    $\Rightarrow x \in \alpha(\boldsymbol{X})$

    or

    $\Rightarrow \exists u.\ u = u_{\mathbf{x}}$ and there is no y s.t. $\boldsymbol{Y} \mid y \in \Sigma'$
    $\Rightarrow \mathcal{A}(f)(u_{\mathbf{x}}) = u_{\mathbf{y}}$
    $\Rightarrow \boldsymbol{X} \in \mathsf{CV}(\Sigma')$ (by def. of $\mathcal{A}$)
    $\Rightarrow$ there is no x s.t. $\boldsymbol{X} \mid x \in \Sigma'$ (by def. of $\mathcal{A}$)
    $\Rightarrow u_{\mathbf{x}} \in \alpha(\boldsymbol{X})$
  * $u' = y$ s.t. $\boldsymbol{Y} - y \in \Sigma'$
    $\Rightarrow \exists u.\ u = x$ s.t. $x \mid f : y \in \Sigma'$
    $\Rightarrow \boldsymbol{X} - x \in \Sigma'$
    (since $\boldsymbol{Y} - y$ can only arise if $\boldsymbol{X} - x \in \Sigma'$)
    $\Rightarrow x \in \alpha(\boldsymbol{X})$

    or

    $\Rightarrow \exists u.\ u = x$ and there is no w s.t. $x \mid f : w \in \Sigma'$
    $\Rightarrow \boldsymbol{X} - x \in \Sigma'$ (by def. of $\mathcal{A}$)
    $\Rightarrow x \in \alpha(\boldsymbol{X})$

or

$\Rightarrow \exists u.\ u = u_{\mathbf{x}}$
$\Rightarrow \boldsymbol{X} \in \mathsf{CV}(\Sigma')$
$\Rightarrow$ there is no x s.t. $\boldsymbol{X} \mid x \in \Sigma'$ (since $u_{\mathbf{x}}$ can only arise if there is no x s.t. $\boldsymbol{X} \mid x \in \Sigma'$)
$\Rightarrow u_{\mathbf{x}} \in \alpha(\boldsymbol{X})$

- $\boldsymbol{X} \mid r : y \in \Sigma'$
  $\Rightarrow \alpha(y) = y$
  $\Rightarrow \alpha(\boldsymbol{X}) = \{x \mid \boldsymbol{X} - x \in \Sigma'\} \cup \{u_{\mathbf{x}}\}$
  or $\alpha(\boldsymbol{X}) = \{x \mid \boldsymbol{X} - x \in \Sigma'\}$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X})$, $y \in \mathcal{D}(r)(u)$ (by definition of $\mathcal{D}$)
- $\boldsymbol{X} \mid \forall r : \boldsymbol{Y} \in \Sigma'$
  $\Rightarrow \alpha(\boldsymbol{X}) = \{x \mid \boldsymbol{X} - x \in \Sigma'\} \cup \{u_{\mathbf{x}}\}$
  or $\alpha(\boldsymbol{X}) = \{x \mid \boldsymbol{X} - x \in \Sigma'\}$
  $\Rightarrow \alpha(\boldsymbol{Y}) = \{y \mid \boldsymbol{Y} - y \in \Sigma'\}$
  or $\alpha(\boldsymbol{Y}) = \{y \mid \boldsymbol{Y} - y \in \Sigma'\} \cup \{u_{\mathbf{y}}\}$
  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X})$
  - $u = u_{\mathbf{x}}$
    $\Rightarrow \mathcal{D}(r)(u_{\mathbf{x}}) = \{y \mid \boldsymbol{X} \mid r : y \in \Sigma'\} = \{y \mid \boldsymbol{Y} - y \in \Sigma'\}$
    $\Rightarrow \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u_{\mathbf{x}})$
  - $u = x$
    $\Rightarrow \mathcal{D}(r)(x) = \{y \mid x \mid r : y \in \Sigma'\}$
    $\cup\{z \mid \boldsymbol{X} \mid r : z \in \Sigma'\}$
    $= \{y \mid \boldsymbol{Y} - y \in \Sigma'\}$ (by conds. 10b, 11)
    $\Rightarrow \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(x)$

  $\Rightarrow$ for each $u \in \alpha(\boldsymbol{X}), \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u)$

□

4.2.3 The Rewrite System

The idea is to provide a rule system that transforms a set of feature constraints into another set of feature constraints that has exactly the properties of the normal form described in definition 4.13. Before we present a rewrite system, we first define how a set of feature clauses is satisfied. We will need definition 4.18 to replace each occurrence of a (big or small) variable in a feature clause by another (big or small) variable. The function **dom** in definition 4.19 is needed to make types as specific as necessary; if a type t has no subtypes t' s.t. for a given feature f, $approp(t, f) \neq approp(t', f)$, then t does not need to be more specific.

Definition 4.17 (Satisfiability of Matrices) *Let Γ be a set of feature clauses, a matrix. Then $\mathcal{I}, \alpha \models \Gamma$ iff for some $\Sigma \in \Gamma$, $\mathcal{I}, \alpha \models \Sigma$.*

Definition 4.18 (Replacement of Variable Occurrences) *Let Σ be a feature clause and let x, y be small variables. Then $\Sigma_{[x/y]}$ stands for Σ with every occurrence of y in Σ replaced by x. If $\boldsymbol{X}$ and $\boldsymbol{Y}$ are big variables, then $\Sigma_{[\mathbf{x}/\mathbf{y}]}$ stands for Σ with every occurrence of $\boldsymbol{Y}$ in Σ replaced by $\boldsymbol{X}$.*

Definition 4.19 (dom) *For every signature $\langle \mathcal{T}, \preceq, \mathcal{F}, \mathcal{R}, \mathrm{approp}\rangle$, **dom** is a total function $\mathcal{T} \times (\mathcal{F} \cup \mathcal{R}) \rightharpoonup \mathcal{P}(\mathcal{T})$ s.t.*

$$\mathbf{dom}(a, f) = max(\{\ b \mid b \preceq a,\ \mathrm{approp}(b, f)\downarrow \text{ and for each } t,\ t' \in \mathcal{V} \text{ s.t. } t \preceq b \text{ and } t' \preceq b, \text{ it is the case that } \mathrm{approp}(t, f) = \mathrm{approp}(t', f)\})$$

Definition 4.20 (Rewrite Rules) *Let Γ be a set of sets of feature constraints, let Σ be a set of feature constraints. Then the rewrite rules are defined as follows:*[12]

For each feature f, for each set-valued attribute r,

1. (a) $\Gamma \uplus \{\Sigma\} \longrightarrow \Gamma \uplus \{\Sigma \cup \{x \mid \top\}\}$
   if x is a main variable in Σ and for no $a \in \mathcal{T}$, $x \mid a \in \Sigma$.

   (b) $\Gamma \uplus \{\Sigma\} \longrightarrow \Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \top\}\}$
   if $\boldsymbol{X}$ is a big variable in Σ and for no $a \in \mathcal{T}$, $\boldsymbol{X} \mid a \in \Sigma$.

2. (a) $\Gamma \uplus \{\Sigma \cup \{x \mid f : y\ ,\ x \mid f : z\}\} \longrightarrow \Gamma \uplus \{\Sigma \cup \{x \mid f : y\ ,\ y \mid z\}\}$
   if $y \neq z$.

   (b) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid f : \boldsymbol{Y}\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Z}\}\} \longrightarrow$
   $\Gamma \uplus \{\Sigma_{[\mathbf{y}/\mathbf{z}]} \cup \{\boldsymbol{X} \mid f : \boldsymbol{Y}\}\}$
   if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ and $\boldsymbol{Y} \neq \boldsymbol{Z}$.

3. (a) i. $\Gamma \uplus \{\Sigma \cup \{x \mid a\ ,\ x \mid b\}\} \longrightarrow \Gamma \uplus \{\Sigma \cup \{x \mid c\}\}$
   if $a \neq b$, glb(a, b) exists and glb$(a, b) = c$.

   ii. $\Gamma \uplus \{\Sigma \cup \{x \mid a\ ,\ x \mid b\}\} \longrightarrow \Gamma$
   if glb(a, b) does not exist.

   (b) i. $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid b\}\} \longrightarrow \Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid c\}\}$
   if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $a \neq b$, glb(a, b) exists and glb$(a, b) = c$.

   ii. $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid b\}\} \longrightarrow \Gamma$
   if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ and glb(a, b) does not exist.

4. (a) $\Gamma \uplus \{\Sigma \cup \{x \mid y\}\} \longrightarrow \Gamma \uplus \{\Sigma_{[x/y]} \cup \{x \mid y\}\}$
   if $x \neq y$ and $y \in \mathsf{FV}(\Sigma)$.

[12] We will use the symbol $\uplus$ for *disjoint union*, which simply ensures that Σ is not contained in Γ. We need this assumption for the termination proof.

(b) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid y\}\} \longrightarrow \Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid y\ ,\ \boldsymbol{X} - y\}\}$
if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ *and* $\boldsymbol{X} - y \notin \Sigma$

(c) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - x\ ,\ \boldsymbol{X} \mid y\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - x\ ,\ \boldsymbol{X} \mid y\ ,\ y \mid x\}\}$
if $x \neq y$ *and* $y \mid x \notin \Sigma$.

5. (a) $\Gamma \uplus \{\Sigma \cup \{x \mid \neg x\}\} \longrightarrow \Gamma$

   (b) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \neg y\ ,\ \boldsymbol{X} - y\}\} \longrightarrow \Gamma$

6. (a) $\Gamma \uplus \{\Sigma \cup \{x \mid a\ ,\ x \mid f : y\}\} \longrightarrow$
   $\Gamma \uplus \{\Sigma \cup \{x \mid c\ ,\ x \mid f : y\} \mid c \in \mathbf{dom}(a, f)\}$
   if $\mathbf{dom}(a, f) \neq \{a\}$.

   (b) $\Gamma \uplus \{\Sigma \cup \{x \mid a\ ,\ x \mid f : y\ ,\ y \mid b\}\} \longrightarrow$
   $\Gamma \uplus \{\Sigma \cup \{x \mid a\ ,\ x \mid f : y\ ,\ y \mid c\}\}$
   if there is a d*, s.t. for all* $t \in \mathcal{V}$ *s.t.* $t \preceq a$*, it is the case that* approp$(t, f)\downarrow$, approp$(t, f) = d$, glb$(d, b)\downarrow$, glb$(d, b) = c$ *and* $c \neq b$.

7. (a) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\}\} \longrightarrow$
   $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid c\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\} \mid c \in \mathbf{dom}(a, f)\}$
   if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ *and* $\mathbf{dom}(a, f) \neq \{a\}$.

   (b) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\ ,\ \boldsymbol{Y} \mid b\}\} \longrightarrow$
   $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\ ,\ \boldsymbol{Y} \mid c\}\}$
   if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ *and there is a* d*, s.t. for all* $t \in \mathcal{V}$ *s.t.* $t \preceq a$*, it is the case that* approp$(t, f)\downarrow$, approp$(t, f) = d$, glb$(d, b)\downarrow$, glb$(d, b) = c$ *and* $c \neq b$.

8. (a) $\Gamma \uplus \{\Sigma \cup \{x \mid a\ ,\ x \mid r : y\}\} \longrightarrow$
   $\Gamma \uplus \{\Sigma \cup \{x \mid c\ ,\ x \mid r : y\} \mid c \in \mathbf{dom}(a, r)\}$
   if $\mathbf{dom}(a, r) \neq \{a\}$.

   (b) $\Gamma \uplus \{\Sigma \cup \{x \mid a\ ,\ x \mid r : y\ ,\ y \mid b\}\} \longrightarrow$
   $\Gamma \uplus \{\Sigma \cup \{x \mid a\ ,\ x \mid r : y\ ,\ y \mid c\}\}$
   if there is a d*, s.t. for all* $t \in \mathcal{V}$ *s.t.* $t \preceq a$*, it is the case that* approp$(t, r)\downarrow$, approp$(t, r) = d$, glb$(d, b)\downarrow$, glb$(d, b) = c$ *and* $c \neq b$.

9. (a) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid r : y\}\} \longrightarrow$
   $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid c\ ,\ \boldsymbol{X} \mid r : y\} \mid c \in \mathbf{dom}(a, r)\}$
   if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ *and* $\mathbf{dom}(a, r) \neq \{a\}$.

   (b) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid r : y\ ,\ y \mid b\}\} \longrightarrow$
   $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid r : y\ ,\ y \mid c\}\}$
   if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ *and there is a* d*, s.t. for all* $t \in \mathcal{V}$ *s.t.* $t \preceq a$*, it is the case that* approp$(t, r)\downarrow$, approp$(t, r) = d$, glb$(d, b)\downarrow$, glb$(d, b) = c$ *and* $c \neq b$.

10. (a) $\Gamma \uplus \{\Sigma \cup \{x \mid \forall r : \boldsymbol{Y} \,,\, x \mid r : z\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{x \mid \forall r : \boldsymbol{Y} \,,\, x \mid r : z \,,\, \boldsymbol{Y} - z\}\}$
if $\boldsymbol{Y} - z \notin \Sigma$.

(b) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \forall r : \boldsymbol{Y} \,,\, \boldsymbol{X} \mid r : z\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \forall r : \boldsymbol{Y} \,,\, \boldsymbol{X} \mid r : z \,,\, \boldsymbol{Y} - z\}\}$
if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ *and* $\boldsymbol{Y} - z \notin \Sigma$.

11. $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \forall r : \boldsymbol{Y} \,,\, z \mid r : z' \,,\, \boldsymbol{X} - z\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \forall r : \boldsymbol{Y} \,,\, z \mid r : z' \,,\, \boldsymbol{X} - z \,,\, \boldsymbol{Y} - z'\}\}$
if $\boldsymbol{Y} - z' \notin \Sigma$.

12. (a) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - y \,,\, \boldsymbol{X} \mid a \,,\, y \mid b\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - y \,,\, \boldsymbol{X} \mid a \,,\, y \mid c\}\}$
if $c \neq b$, $glb(a,b)\downarrow$ *and* $glb(a,b) = c$.

(b) $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - y \,,\, \boldsymbol{X} \mid a \,,\, y \mid b\}\} \longrightarrow \Gamma$
if $glb(a,b)$ *does not exist.*

13. $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - y \,,\, \boldsymbol{X} \mid f : \boldsymbol{Z} \,,\, y \mid f : z'\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - y \,,\, \boldsymbol{X} \mid f : \boldsymbol{Z} \,,\, y \mid f : z' \,,\, \boldsymbol{Z} - z'\}\}$
if $\boldsymbol{Z} - z' \notin \Sigma$.

14. $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - x \,,\, \boldsymbol{X} \mid r : z \,,\, x \mid \forall r : \boldsymbol{Y}\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - x \,,\, \boldsymbol{X} \mid r : z \,,\, x \mid \forall r : \boldsymbol{Y} \,,\, \boldsymbol{Y} - z\}\}$
if $\boldsymbol{Y} - z \notin \Sigma$.

Now we have to prove that our rewrite rules produce a normal form, that they preserve the denotation of a feature clause and, finally, that the process of rewriting terminates after a finite number of steps.[13]

Proposition 4.7 *For each set* Σ *of feature constraints,* Σ *is in normal form iff* $\{\Sigma\} \not\longrightarrow$.

Proof. Since each normal form condition in defintion 4.13 has one corresponding rewrite rule, we can see, by an easy case analysis, that if a rewrite rule applies then the corresponding normal form condition is violated and, conversely, if a constraint system is *not* in normal form (i.e., if at least one of the normal form conditions is not met) then there is always an applicable (namely the corresponding) rewrite rule.

$\square$

Now we need to show that the rewrite system preserves the denotation of a matrix.

[13] The following proofs are based on (Götz prep).

Proposition 4.8 *For each matrix* Γ, *for each matrix* Γ', *for each interpretation* $\langle \mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{D} \rangle$, *for each variable assignment* α, *if* $\Gamma \longrightarrow \Gamma'$, *then* $\mathcal{I}, \alpha \models \Gamma$ *iff* $\mathcal{I}, \alpha \models \Gamma'$.

Proof.

1. $\{x \mid \top\}$ and $\{\boldsymbol{X} \mid \top\}$ can always be added to a matrix since $\mathcal{I}, \alpha \models x \mid \top$ and $\mathcal{I}, \alpha \models \boldsymbol{X} \mid \top$ is always true.

2. (a) i. $\mathcal{I}, \alpha \models \Sigma \cup \{x \mid f : y \ , \ x \mid f : z\}$, $y \neq z$ and $y \mid z \notin \Sigma$.

   $\Leftrightarrow \mathcal{I}, \alpha \models \Sigma$ and $\mathcal{A}(f)(\alpha(x)) \downarrow, \mathcal{A}(f)(\alpha(x)) = \alpha(y)$, $\mathcal{A}(f)(\alpha(x)) = \alpha(z)$

   $\Leftrightarrow \mathcal{A}(f)(\alpha(x)) \downarrow$, $\mathcal{A}(f)(\alpha(x)) = \alpha(y)$, $\alpha(y) = \alpha(z)$

   $\Leftrightarrow \mathcal{I}, \alpha \models \Sigma \cup \{x \mid f : y, \ y \mid z\}$

   ii. $\mathcal{I}, \alpha \models \Sigma \cup \{x \mid f : y, \ x \mid f : z\}$, $y \neq z$ and $y \mid z \in \Sigma$.

   $\Leftrightarrow \mathcal{I}, \alpha \models \Sigma$, $y \mid z \in \Sigma$, $\alpha(y) = \alpha(z)$ $\mathcal{A}(f)(\alpha(x)) \downarrow$, $\mathcal{A}(f)(\alpha(x)) = \alpha(y)$, $\mathcal{A}(f)(\alpha(x)) = \alpha(z)$

   $\Leftrightarrow \mathcal{I}, \alpha \models \Sigma \cup \{x \mid f : y\}$

   (b) $\mathcal{I}, \alpha \models \Sigma \cup \{\boldsymbol{X} \mid f : \ \boldsymbol{Y} \ , \ \boldsymbol{X} \mid f : \ \boldsymbol{Z}\}$, $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ and $\boldsymbol{Y} \neq \boldsymbol{Z}$

   $\Leftrightarrow \mathcal{I}, \alpha \models \Sigma$ and $\forall u \in \alpha(\boldsymbol{X}).\mathcal{A}(f)(u) \downarrow, \mathcal{A}(f)\alpha(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$, $\alpha(\boldsymbol{Z}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$ and $\alpha(\boldsymbol{X}) \neq \emptyset$

   $\Leftrightarrow \mathcal{I}, \alpha \models \Sigma$ and $\forall u \in \alpha(\boldsymbol{X}).\mathcal{A}(f)(u) \downarrow, \mathcal{A}(f)\alpha(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$, $\alpha(\boldsymbol{Z}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$, $\alpha(\boldsymbol{X}) \neq \emptyset$ and $\alpha(\boldsymbol{Y}) = \alpha(\boldsymbol{Z})$

   $\Leftrightarrow \mathcal{I}, \alpha \models \Sigma_{[\mathbf{y}/\mathbf{z}]} \cup \{\boldsymbol{X} \mid f : \ \boldsymbol{Y}\}$

3. (a) i. $\mathcal{I}, \alpha \models \{x \mid a \ , \ x \mid b\}$, $glb(a, b) \downarrow$, $glb(a, b) = c$ [1ex] $\Leftrightarrow \mathcal{S}(\alpha(x)) \preceq a, \mathcal{S}(\alpha(x)) \preceq b$, $glb(a, b) \downarrow$, $glb(a, b) = c$

   $\Leftrightarrow \mathcal{S}(\alpha(x)) \preceq glb(a, b) = c$

   $\Leftrightarrow \mathcal{I}, \alpha \models \{x \mid c\}$

   ii. If $glb(a, b)$ does not exist, then $\not\models \{x \mid a, x \mid b\}$ and hence $\Sigma \cup \{x \mid a \ , \ x \mid b\}$ is not satisfiable.

   (b) i. $\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a \ , \ \boldsymbol{X} \mid b\}$, $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $glb(a, b) \downarrow$, $glb(a, b) = c$

   $\Leftrightarrow$ for each $u \in \alpha(\boldsymbol{X}), \mathcal{S}(u) \preceq a$ and $\mathcal{S}(u) \preceq b$, $glb(a, b) \downarrow$, $glb(a, b) = c$

   $\Leftrightarrow$ for each $u \in \alpha(\boldsymbol{X}), \mathcal{S}(u) \preceq a$ and $\mathcal{S}(u) \preceq b$, $glb(a, b) \downarrow$,

$glb(a, b) = c$ and $\mathcal{S}(u) \preceq glb(a, b) = c$

$\Leftrightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid c\}$

ii. If $glb(a, b)$ does not exist, then $\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a, \boldsymbol{X} \mid b\}$ only if $\alpha(\boldsymbol{X}) = \emptyset$. But, since $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $\alpha(\boldsymbol{X}) \neq \emptyset$, therefore $\not\models \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid b\}$ and hence $\Sigma \cup \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid b\}$ s.t. $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, is not satisfiable.

4. (a) $\mathcal{I}, \alpha \models \Sigma \cup \{x \mid y\}$, $y \neq x$, $y \in \mathsf{FV}(\Sigma)$

$\Leftrightarrow \mathcal{I}, \alpha \models \Sigma$ and $\alpha(x) = \alpha(y)$

$\Leftrightarrow \mathcal{I}, \alpha \models \Sigma_{[x/y]} \cup \{x \mid y\}$

(b) $\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid y\}$ and $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$

$\Leftrightarrow \mathcal{I}, \alpha \models \Sigma$, for each $u \in \alpha(\boldsymbol{X})$, $u = \alpha(y)$ and $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$

$\Leftrightarrow \mathcal{I}, \alpha \models \Sigma$, for each $u \in \alpha(\boldsymbol{X})$, $u = \alpha(y)$ and $\alpha(\boldsymbol{X}) \neq \emptyset$ (by def. of $\mathsf{CV}(\Sigma)$)

$\Leftrightarrow \mathcal{I}, \alpha \models \Sigma \cup \{\boldsymbol{X} \mid y\ ,\ \boldsymbol{X} - y\}$

(c) $\mathcal{I}, \alpha \models \{\boldsymbol{X} - x\ ,\ \boldsymbol{X} \mid y\}$

$\Leftrightarrow \alpha(x) \in \alpha(\boldsymbol{X})$ and for each $u \in \alpha(\boldsymbol{X}), u = \alpha(y)$

$\Leftrightarrow \alpha(x) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X}), u = \alpha(y)$ and $\alpha(y) = \alpha(x)$

$\Leftrightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} - x\ ,\ \boldsymbol{X} \mid y\ ,\ y \mid x\}$

5. (a) Clearly, $\{x \mid \neg x\}$ is not satisfiable and hence, there is no $\mathcal{I}$ and α, s.t. $\mathcal{I}, \alpha \models \Sigma \cup \{x \mid \neg x\}$.

(b) $\alpha(y)$ cannot be both a member and not a member of $\alpha(\boldsymbol{X})$. Thus, $\Sigma \cup \{\boldsymbol{X} \mid \neg y\ ,\ \boldsymbol{X} - y\}$ is not satisfiable.

6. (a) We need to show that
$\mathcal{I}, \alpha \models \{x \mid a\ ,\ x \mid f : y\}$

$\Leftrightarrow \exists c \in \mathbf{dom}(a, f).\ \mathcal{I}, \alpha \models \{x \mid c\ ,\ x \mid f : y\}$
Right to left is obvious, since for each $c \in \mathbf{dom}(a, f)$, $c \preceq a$.
$\mathcal{I}, \alpha \models \{x \mid a\ ,\ x \mid f : y\}$

$\Rightarrow \mathcal{S}(\alpha(x)) \preceq a,\ \mathcal{A}(f)(\alpha(x)) \downarrow,\ \mathcal{A}(f)(\alpha(x)) = \alpha(y)$

$\Rightarrow \exists c \in \mathbf{dom}(a, f).\ \mathcal{S}(\alpha(x)) \preceq c,\ \mathcal{A}(f)(\alpha(x)) \downarrow,$
$\mathcal{A}(f)(\alpha(x)) = \alpha(y)$

$\Rightarrow \exists c \in \mathbf{dom}(a, f).\ \mathcal{I}, \alpha \models \{x \mid c\ ,\ x \mid f : y\}$

(b) Again, right to left is obvious, since $c \preceq b$. For left to right, we have:

$\mathcal{I}, \alpha \models \{x \mid a\ ,\ x \mid f : y\ ,\ y \mid b\},\ \exists d.\ \forall t \preceq a.\ approp(t, f) \downarrow,$
$approp(t, f) = d,\ glb(b, d) \downarrow$ and $glb(b, d) = c$

$\Rightarrow \mathcal{S}(\alpha(x)) \preceq a,\ \mathcal{A}(f)(\alpha(x)) \downarrow,\ \mathcal{A}(f)(\alpha(x)) = \alpha(y),$
$\mathcal{S}(\alpha(y)) \preceq b,\ \mathcal{S}(\alpha(y)) \preceq d,\ glb(b, d) \downarrow$ and $glb(b, d) = c$

$\Rightarrow \mathcal{S}(\alpha(x)) \preceq a,\ \mathcal{A}(f)(\alpha(x)) \downarrow,\ \mathcal{A}(f)(\alpha(x)) = \alpha(y)$ and $\mathcal{S}(\alpha(y)) \preceq c$

$\Rightarrow \mathcal{I}, \alpha \models \{x \mid a\ ,\ x \mid f : y\ ,\ y \mid c\}$

7. (a) We need to show that
$\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\}$ and $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$

$\Leftrightarrow \exists c \in \mathbf{dom}(a, f).\ \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid c\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\}$
Right to left is obvious, since for each $c \in \mathbf{dom}(a, f),\ c \preceq a$.
$\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\}$ and $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$

$\Rightarrow$ For each $u \in \alpha(\boldsymbol{X}),\ \mathcal{S}(u) \preceq a,\ \mathcal{A}(f)(u) \downarrow,\ \alpha(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$ and $\alpha(\boldsymbol{X}) \neq \emptyset$

$\Rightarrow \exists c \in \mathbf{dom}(a, f).\ \forall u \in \alpha(\boldsymbol{X}).\mathcal{S}(u) \preceq c,\ \mathcal{A}(f)(u) \downarrow,$
$\alpha(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$ and $\alpha(\boldsymbol{X}) \neq \emptyset$

$\Rightarrow \exists c \in \mathbf{dom}(a, f).\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid c\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\}$

(b) Again, right to left is obvious, since $c \preceq b$. For left to right, we have:
$\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\ ,\ \boldsymbol{Y} \mid b\}, \exists d.\ \forall t \preceq a.\ approp(t, f) \downarrow,$
$approp(t, f) = d,\ glb(b, d) \downarrow$ and $glb(b, d) = c$ and $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$.

$\Rightarrow$ For each $u \in \alpha(\boldsymbol{X}),\ \mathcal{S}(u) \preceq a,\ \mathcal{A}(f)(u) \downarrow,$
$\alpha(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$ and $\alpha(\boldsymbol{X}) \neq \emptyset,$
for each $u' \in \alpha(\boldsymbol{Y}),\ \mathcal{S}(u') \preceq b, \mathcal{S}(u') \preceq d, glb(b, d) \downarrow$ and $glb(b, d) = c$ and $\alpha(\boldsymbol{Y}) \neq \emptyset$

$\Rightarrow$ For each $u \in \alpha(\boldsymbol{X}),\ \mathcal{S}(u) \preceq a,\ \mathcal{A}(f)(u) \downarrow,$
$\alpha(\boldsymbol{Y}) = \{\mathcal{A}(f)(u) \mid u \in \alpha(\boldsymbol{X})\}$ and $\alpha(\boldsymbol{X}) \neq \emptyset,$
for each $u' \in \alpha(\boldsymbol{Y}), \mathcal{S}(u') \preceq b, \mathcal{S}(u') \preceq d, \mathcal{S}(u') \preceq c, glb(b, d) \downarrow,$
$glb(b, d) = c$ and $\alpha(\boldsymbol{Y}) \neq \emptyset$

$\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid f : \boldsymbol{Y}\ ,\ \boldsymbol{Y} \mid c\}$

8. (a) We need to show that
$\mathcal{I}, \alpha \models \{x \mid a\ ,\ x \mid r : y\}$
$\Leftrightarrow \exists c \in \mathbf{dom}(a, r).\ \mathcal{I}, \alpha \models \{x \mid c\ ,\ x \mid r : y\}$
Right to left is obvious, since for each $c \in \mathbf{dom}(a, r),\ c \preceq a$.
$\mathcal{I}, \alpha \models \{x \mid a\ ,\ x \mid r : y\}$

$\Rightarrow \mathcal{S}(\alpha(x)) \preceq a,\ \alpha(y) \in \mathcal{D}(r)(\alpha(x))$

$\Rightarrow \exists c \in \mathbf{dom}(a,r).\ \mathcal{S}(\alpha(x)) \preceq c,\ \alpha(y) \in \mathcal{D}(r)(\alpha(x))$

$\Rightarrow \exists c \in \mathbf{dom}(a,r).\ \mathcal{I}, \alpha \models \{x \mid c, x \mid r : y\}$

(b) Again, right to left is obvious, since $c \preceq b$. For left to right, we have:

$\mathcal{I}, \alpha \models \{x \mid a\ ,\ x \mid r : y\ ,\ y \mid b\},\ \exists d.\ \forall t \preceq a.\ approp(t,r) \downarrow,$ $approp(t,r) = d,\ glb(b,d) \downarrow$ and $glb(b,d) = c$

$\Rightarrow \mathcal{S}(\alpha(x)) \preceq a,\ \alpha(y) \in \mathcal{D}(r)(\alpha(x)),$
$\mathcal{S}(\alpha(y)) \preceq b,\ \mathcal{S}(\alpha(y)) \preceq d,\ glb(b,d) \downarrow$ and $glb(b,d) = c$

$\Rightarrow \mathcal{S}(\alpha(x)) \preceq a,\ \alpha(y) \in \mathcal{D}(r)(\alpha(x))$ and $\mathcal{S}(\alpha(y)) \preceq c$

$\Rightarrow \mathcal{I}, \alpha \models \{x \mid a\ ,\ x \mid r : y\ ,\ y \mid c\}$

9. (a) We need to show that
$\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid r : y\}$ and $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$

$\Leftrightarrow \exists c \in \mathbf{dom}(a,r).\ \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid c\ ,\ \boldsymbol{X} \mid r : y\}$
Right to left is obvious, since for each $c \in \mathbf{dom}(a,r),\ c \preceq a$.
$\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid r : y\}$ and $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$

$\Rightarrow$ For each $u \in \alpha(\boldsymbol{X})$, $\mathcal{S}(u) \preceq a$, $\alpha(y) \in \mathcal{D}(r)(u)$
and $\alpha(\boldsymbol{X}) \neq \emptyset$

$\Rightarrow \exists c \in \mathbf{dom}(a,r).\ \forall u \in \alpha(\boldsymbol{X}).\mathcal{S}(u) \preceq c,\ \alpha(y) \in \mathcal{D}(r)(u)$
and $\alpha(\boldsymbol{X}) \neq \emptyset$

$\Rightarrow \exists c \in \mathbf{dom}(a,r).\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid c\ ,\ \boldsymbol{X} \mid r : y\}$

(b) Again, right to left is obvious, since $c \preceq b$. For left to right, we have:

$\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid r : y\ ,\ y \mid b\}, \exists d.\ \forall t \preceq a.\ approp(t,r) \downarrow,$ $approp(t,r) = d,\ glb(b,d) \downarrow$ and $glb(b,d) = c$ and $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$.

$\Rightarrow$ For each $u \in \alpha(\boldsymbol{X})$, $\mathcal{S}(u) \preceq a, \alpha(y) \in \mathcal{D}(r)(u)$,
$\alpha(\boldsymbol{X}) \neq \emptyset$, $\mathcal{S}(\alpha(y)) \preceq b, \mathcal{S}(\alpha(y)) \preceq d$, $glb(b,d) \downarrow$
and $glb(b,d) = c$

$\Rightarrow$ For each $u \in \alpha(\boldsymbol{X})$, $\mathcal{S}(u) \preceq a,\ \alpha(y) \in \mathcal{D}(r)(u)$,
$\alpha(\boldsymbol{X}) \neq \emptyset$,
$\mathcal{S}(\alpha(y)) \preceq b, \mathcal{S}(\alpha(y)) \preceq d, \mathcal{S}(\alpha(y)) \preceq c$, $glb(b,d) \downarrow$,
$glb(b,d) = c$ and $\alpha(\boldsymbol{X}) \neq \emptyset$

$\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid a\ ,\ \boldsymbol{X} \mid r : y\ ,\ y \mid c\}$

10. (a) $\mathcal{I}, \alpha \models \{x \mid \forall r : \boldsymbol{Y}\ ,\ x \mid r : z\}$

$\Leftrightarrow \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(\alpha(x))$ and $\alpha(z) \in \mathcal{D}(r)(\alpha(x))$

$\Leftrightarrow \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(\alpha(x)), \alpha(z) \in \mathcal{D}(r)(\alpha(x))$
and $\alpha(z) \in \alpha(\boldsymbol{Y})$

$\Leftrightarrow \mathcal{I}, \alpha \models \{x \mid \forall r : \boldsymbol{Y} \ , x \mid r : z \ , \ \boldsymbol{Y} - z\}$

(b) $\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid \forall r : \boldsymbol{Y} \ , \ \boldsymbol{X} \mid r : z\}$ and $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$

$\Leftrightarrow$ For each $u \in \alpha(\boldsymbol{X}), \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u)$ and $\alpha(z) \in \mathcal{D}(r)(u)$

$\Leftrightarrow$ For each $u \in \alpha(\boldsymbol{X}), \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u), \alpha(z) \in \mathcal{D}(r)(u)$ and $\alpha(z) \in \alpha(\boldsymbol{Y})$

$\Leftrightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid \forall r : \boldsymbol{Y} \ , \ \boldsymbol{X} \mid r : z \ , \ \boldsymbol{Y} - z\}$

11. $\mathcal{I}, \alpha \models \{\boldsymbol{X} \mid \forall r : \boldsymbol{Y} \ , \ z \mid r : z' \ , \ \boldsymbol{X} - z\}$

$\Leftrightarrow$ for each $u \in \alpha(\boldsymbol{X}), \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u)$, $\alpha(z') \in \mathcal{D}(r)(\alpha(z))$ and $\alpha(z) \in \alpha(\boldsymbol{X})$

$\Leftrightarrow$ for each $u \in \alpha(\boldsymbol{X}), \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u)$, $\alpha(z') \in \mathcal{D}(r)(\alpha(z))$, $\alpha(z) \in \alpha(\boldsymbol{X})$ and $\alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(\alpha(z))$

$\Leftrightarrow$ for each $u \in \alpha(\boldsymbol{X}), \alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(u)$, $\alpha(z') \in \mathcal{D}(r)(\alpha(z))$, $\alpha(z) \in \alpha(\boldsymbol{X})$, $\alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(\alpha(z))$ and $\alpha(z') \in \alpha(\boldsymbol{Y})$

$\Leftrightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid \forall r : \boldsymbol{Y} \ , \ z \mid r : z' \ , \ \boldsymbol{X} - z \ , \ \boldsymbol{Y} - z'\}$

12. (a) $\mathcal{I}, \alpha \models \{\boldsymbol{X} - y \ , \ \boldsymbol{X} \mid a \ , \ y \mid b\}$ $glb(a,b)\downarrow$ and $glb(a,b) = c$.

$\Leftrightarrow \alpha(y) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X}), \mathcal{S}(u) \preceq a$, $\mathcal{S}(\alpha(y)) \preceq b$, $glb(a,b)\downarrow$ and $glb(a,b) = c$.

$\Leftrightarrow \alpha(y) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X}), \mathcal{S}(u) \preceq a$, $\mathcal{S}(\alpha(y)) \preceq b$ and $\mathcal{S}(\alpha(y)) \preceq a$, $glb(a,b)\downarrow$ and $glb(a,b) = c$.

$\Leftrightarrow \alpha(y) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X}), \mathcal{S}(u) \preceq a$ and $\mathcal{S}(\alpha(y)) \preceq glb(a,b) = c$.

$\Leftrightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} - y \ , \ \boldsymbol{X} \mid a \ , \ y \mid c\}$

(b) If $glb(a,b)$ does not exist, then $\not\models \{\boldsymbol{X} - y \ , \ \boldsymbol{X} \mid a \ , \ y \mid b\}$: It is not possible that $\mathcal{S}(\alpha(y)) \preceq a$ and $\mathcal{S}(\alpha(y)) \preceq b$ if $glb(a,b)\uparrow$, but $\boldsymbol{X} - y$ requires $\alpha(y) \in \alpha(\boldsymbol{X})$.
Hence $\Sigma \cup \{\boldsymbol{X} - y \ , \ \boldsymbol{X} \mid a \ , \ y \mid b\}$ is not satisfiable if $glb(a,b)\uparrow$.

13. $\mathcal{I}, \alpha \models \{\boldsymbol{X} - y \ , \ \boldsymbol{X} \mid f : \boldsymbol{Z} \ , \ y \mid f : z'\}$

$\Leftrightarrow \alpha(y) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X}), \mathcal{A}(f)(u)\downarrow$ and $\mathcal{A}(f)(u) \in \alpha(\boldsymbol{Z})$, $\mathcal{A}(f)(\alpha(y))\downarrow$ and $\mathcal{A}(f)(\alpha(y)) = \alpha(z')$

$\Leftrightarrow \alpha(y) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X}), \mathcal{A}(f)(u)\downarrow$ and $\mathcal{A}(f)(u) \in \alpha(\boldsymbol{Z})$, $\mathcal{A}(f)(\alpha(y))\downarrow$, $\mathcal{A}(f)(\alpha(y)) = \alpha(z')$ and $\alpha(z') \in \alpha(\boldsymbol{Z})$

$\Leftrightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} - y \ , \ \boldsymbol{X} \mid f : \boldsymbol{Z} \ , \ y \mid f : z' \ , \ \boldsymbol{Z} - z'\}$

14. $\mathcal{I}, \alpha \models \{\boldsymbol{X} - x \ , \ \boldsymbol{X} \mid r : z \ , x \mid \forall r : \boldsymbol{Y}\}$

$\Leftrightarrow \alpha(x) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X})$, $\alpha(z) \in \mathcal{D}(r)(u)$

and $\alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(\alpha(x))$
$\Leftrightarrow \alpha(x) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X})$, $\alpha(z) \in \mathcal{D}(r)(u)$,
$\alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(\alpha(x))$ and $\alpha(z) \in \mathcal{D}(r)(\alpha(x))$

$\Leftrightarrow \alpha(x) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X})$, $\alpha(z) \in \mathcal{D}(r)(u)$,
$\alpha(\boldsymbol{Y}) \supseteq \mathcal{D}(r)(\alpha(x))$, $\alpha(z) \in \mathcal{D}(r)(\alpha(x))$
and $\alpha(z) \in \alpha(\boldsymbol{Y})$

$\Leftrightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} - x\ ,\ \boldsymbol{X} \mid r : z\ ,\ x \mid \forall r :\ \boldsymbol{Y}\ ,\ \boldsymbol{Y} - z\}$

$\square$

It remains to be shown that the rewriting process terminates.

Proposition 4.9 (Termination) *If Γ is a finite matrix of finite sets, then there are Γ', Γ'' and natural numbers n and m s.t. $\Gamma \xrightarrow{n} \Gamma' \xrightarrow{m} \Gamma'' \not\longrightarrow$, where rule 1) does not apply to Γ'.*

Proof. Suppose $\Gamma = \{\Sigma_1, ..., \Sigma_k\}$. Γ' can be derived in at most $2 \times (|\Sigma_1| \times ... \times |\Sigma_k|)$ steps, since for each i, $|main(\Sigma_i) \cup \mathsf{BV}(\Sigma_i)| \leq 2\ \times |\Sigma_i|$. Since none of the other rules introduce new (big or small) variables, and no rule deletes constraints of the form $x \mid a$ or $\boldsymbol{X} \mid a$ (note that rules $3a(i)$ and $3b(i)$ only become applicable if there are, for some x or $\boldsymbol{X}$, at least two constraints of the form $x \mid a$ or $\boldsymbol{X} \mid a$), we can consider the rest of the rules without rule 1.

We proceed by associating a weight with each set of constraints. Unlike (Götz prep), we will use both positive and negative weights. Negative weights will be used for feature constraints which are *added* to a feature clause.

Each rule application generates from a set Σ (which is removed from the matrix) finitely many sets of constraints Σ'. Thus, it suffices to show that for each of those new Σ', $0 \leq \mathit{weight}(\Sigma') < \mathit{weight}(\Sigma)$, i.e., that a weight of a set of constraints decreases but it can never become negative. A derivation defines a finitely branching tree, and, in order to show that the tree is finite, we need to show that every branch is finite.

Again, f is a feature and r is a set-valued attribute.

- $\mathit{weight}(x \mid a) = |\{b \mid b \preceq a\}|$
- $\mathit{weight}(\boldsymbol{X} \mid a) = |\{b \mid b \preceq a\}|$
- $\mathit{weight}(x \mid y) = -1$
- $\mathit{weight}(\boldsymbol{X} \mid y) = -1$
- $\mathit{weight}(\boldsymbol{X} - y) = -1$
- $\mathit{weight}(x \mid \neg y) = 1$

- $weight(x \mid f : y) = 2$
- $weight(x \mid r : y) = 2$
- $weight(x \mid \forall r : \boldsymbol{Y}) = 2$
- $weight(\boldsymbol{X} \mid \neg y) = 1$
- $weight(\boldsymbol{X} \mid f : \boldsymbol{Y}) = 2$
- $weight(\boldsymbol{X} \mid r : y) = 2$
- $weight(\boldsymbol{X} \mid \forall r : \boldsymbol{Y}) = 2$
- $weight(\Sigma) = \left(\sum_{\sigma \in \Sigma} weight(\sigma)\right)$

$$+ |\{x|y \mid x \in main(\Sigma), y \in \mathsf{FV}(\Sigma)\}|$$
$$+ |\{\boldsymbol{X}|y \mid \boldsymbol{X} \in \mathsf{BV}(\Sigma), y \in main(\Sigma)\}|$$
$$+ |\{\boldsymbol{X} - y \mid \boldsymbol{X} \in \mathsf{BV}(\Sigma), y \in main(\Sigma)\}|$$

It can easily be seen that the weight of a set of constraints, though decreasing, can never become smaller than 0: Clearly, the set of all feature constraints $x \mid y$, $\boldsymbol{X} \mid y$, and $\boldsymbol{X} - y$ in Σ is always a subset of $\{x|y \mid x \in main(\Sigma), y \in \mathsf{FV}(\Sigma)\} \cup \{\boldsymbol{X}|y \mid \boldsymbol{X} \in \mathsf{BV}(\Sigma), y \in main(\Sigma)\} \cup \{\boldsymbol{X} - y \mid \boldsymbol{X} \in \mathsf{BV}(\Sigma), y \in main(\Sigma)\}$. But the cardinality of these three sets is added to the amount of component weights of all $\sigma \in \Sigma$. It is evident that the number of the negative weights cannot become bigger than the cardinality of the three additional sets and, consequently, the weight of Σ cannot become negative.

Now consider rule application. Let Σ' arise from Σ by application of a rewrite rule.

- Suppose rule $2a$ is applied. We know that $|main(\Sigma')| \leq |main(\Sigma)|$ and $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$. Then $weight(\Sigma') \leq weight(\Sigma) - 3$, if $x \mid f : z$ is replaced by $y \mid z$, or $weight(\Sigma') \leq weight(\Sigma) - 2$, if $y \mid z$ is already a member of Σ. Thus, $weight(\Sigma') < weight(\Sigma)$.

- If rule $2b$ is applied to Σ, then $weight(\Sigma') \leq weight(\Sigma) - 2$, since $main(\Sigma') = main(\Sigma)$, $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma) - 1$ and hence, $weight(\Sigma') < weight(\Sigma)$.

- Suppose rule $3a(i)$ is applied: $\Gamma \uplus \{\Sigma \cup \{x \mid a \ , x \mid b\}\} \longrightarrow \Gamma \uplus \{\Sigma \cup \{x \mid c\}\}$, where $a \neq b$, $glb(a, b) \downarrow$ and $glb(a, b) = c$. Let $A = \{a' \mid a' \preceq a\}$, $B = \{b' \mid b' \preceq b\}$ and $C = \{c' \mid c' \preceq c\}$. Assume w.l.o.g. that $|A| \geq |B|$. Since $a \neq b$ and $glb(a,b) \downarrow$, we know that $|A| \geq 2$. Thus, $weight(x \mid a) + weight(x \mid b) = |A| + |B| \geq 2 + |B| > |B| \geq |C| = weight(x \mid c)$. Since $main(\Sigma') = main(\Sigma)$ and $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$, $weight(\Sigma \cup \{x \mid c\}) < weight(\Sigma \cup \{x \mid a\ ,\ x \mid b\})$.

- If rule $3a(ii)$ is applied, we are at a terminal node of the tree.
- The weight of Σ after application of rule $3b(i)$ corresponds to the weight of Σ after application of rule 3a, since $main(\Sigma') = main(\Sigma)$, $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$ and $weight(\boldsymbol{X} \mid a) = weight(x \mid a)$. Hence we have $weight(\Sigma \cup \{\boldsymbol{X} \mid c\}) < weight(\Sigma \cup \{\boldsymbol{X} \mid a, \boldsymbol{X} \mid b\})$.
- If rule $3b(ii)$ is applied, we are at a terminal node of the tree.
- Suppose rule $4a$ applies:
  $\Gamma \uplus \{\Sigma \cup \{x \mid y\}\} \longrightarrow \Gamma \uplus \{\Sigma_{[x/y]} \cup \{x \mid y\}\}$ with $x \neq y$ and $y \in \mathsf{FV}(\Sigma)$. Then $\sum\limits_{\sigma \in (\Sigma \cup \{x|y\})} weight(\sigma) = \sum\limits_{\sigma \in (\Sigma_{[x/y]} \cup \{x|y\})} weight(\sigma)$, but $|main(\Sigma \cup \{x \mid y\})| = |main(\Sigma_{[x/y]} \cup \{x \mid y\})| - 1$. Thus, $|\{x|y \mid x \in main(\Sigma), y \in \mathsf{FV}(\Sigma)\}| < |\{x|y \mid x \in main(\Sigma_{[x/y]}), y \in \mathsf{FV}(\Sigma_{[x/y]})\}|$, $|\{\boldsymbol{X}|y \mid \boldsymbol{X} \in \mathsf{BV}(\Sigma), y \in main(\Sigma)\}| \leq |\{\boldsymbol{X}|y \mid \boldsymbol{X} \in \mathsf{BV}(\Sigma_{[x/y]}), y \in main(\Sigma_{[x/y]})\}|$, $|\{\boldsymbol{X} - y \mid \boldsymbol{X} \in \mathsf{BV}(\Sigma), y \in main(\Sigma)\}| \leq |\{\boldsymbol{X} - y \mid \boldsymbol{X} \in \mathsf{BV}(\Sigma_{[x/y]}), y \in main(\Sigma_{[x/y]})\}|$. Thus, $weight(\Sigma_{[x/y]} \cup \{x \mid y\}) < weight(\Sigma \cup \{x \mid y\})$.
- If rule $4b$ is applied, then $\sum\limits_{\sigma' \in \Sigma'} weight(\sigma') = (\sum\limits_{\sigma \in \Sigma} weight(\sigma)) - 1$ and, since $main(\Sigma') = main(\Sigma)$ and $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$, $weight(\Sigma') < weight(\Sigma)$.
- If rule $4c$ is applied, then $\sum\limits_{\sigma' \in \Sigma'} weight(\sigma') = (\sum\limits_{\sigma \in \Sigma} weight(\sigma)) - 1$ and, since $main(\Sigma') = main(\Sigma)$ and $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$, $weight(\Sigma') < weight(\Sigma)$.
- If rule $5a$ is applied, we are at a terminal node of the tree.
- Again, if rule $5b$ is applied, we are at a terminal node of the tree.
- Suppose rule $6a$ is applied, where, for some set of constraints Φ, $\Sigma = \Phi \cup \{x \mid a\ ,\ x \mid f : y\}$ and $\Sigma' = \Phi \cup \{x \mid c\ ,\ x \mid f : y\}$. Then we know that for each $c \in \mathbf{dom}(a, f)$, where $\mathbf{dom}(a, f) \neq \{a\}$, $c \preceq a$ and $c \neq a$.
  Thus, $weight(x|c) < weight(x|a)$, and, since $main(\Sigma') = main(\Sigma)$ and $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$, $weight(\Sigma') < weight(\Sigma)$.

- If rule $6b$ is applied, where, for some set of constraints Φ, $\Sigma = \Phi \cup \{x \mid a\ ,\ x \mid f : y\ ,\ y \mid b\}$ and $\Sigma' = \Phi \cup \{x \mid a\ ,\ x \mid f : y\ ,\ y \mid c\}$, then we know that, if there is a d s.t. for all $t \in \mathcal{V}$, s.t. $t \preceq a$, it is the case that $approp(t, f) \downarrow$, $approp(t, f) = d$, $glb(d, b) \downarrow$, $glb(d, b) = c$ and $c \neq b$.

  Thus, $weight(y|c) < weight(y|b)$, and, since $main(\Sigma') = main(\Sigma)$ and $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$, $weight(\Sigma') < weight(\Sigma)$.

- Rules $7a$, $8a$ and $9a$ reduce the weight of a feature clause in the same way rule $6a$ does. The reduced weights after application of rules $7b$, $8b$ and $9b$ correspond to the weights after applicaton of rule $6b$.

- If rule $10a$, $10b$ or 11 is applied, then $\sum\limits_{\sigma' \in \Sigma'} weight(\sigma') = (\sum\limits_{\sigma \in \Sigma} weight(\sigma)) - 1$ and, since $main(\Sigma') = main(\Sigma)$ and $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$, $weight(\Sigma') < weight(\Sigma)$.

- Suppose rule $12a$ is applied, where, for some set of constraints Φ, $\Sigma = \Phi \cup \{\boldsymbol{X} - y\ ,\ \boldsymbol{X} \mid a\ ,\ y \mid b\}$ and $\Sigma' = \Phi \cup \{\boldsymbol{X} - y\ ,\ \boldsymbol{X} \mid a,\ y \mid c\}$, s.t. $c \neq b$, $glb(a, b) \downarrow$ and $glb(a, b) = c$.
  Let $A = \{a' \mid a' \preceq a\}$, $B = \{b' \mid b' \preceq b\}$ and $C = \{c' \mid c' \preceq c\}$. Since $c \neq b$ and $glb(a, b) \downarrow$, we know that $c \prec b$, and hence $|C| < |B|$. Thus, $weight(y \mid c) < weight(y \mid b)$. Since $main(\Sigma') = main(\Sigma)$ and $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$, $weight(\Sigma') < weight(\Sigma)$.

- If rule 12b is applied, we are at a terminal node of the tree.

- If rule 13 or 14 is applied, then $\sum\limits_{\sigma' \in \Sigma'} weight(\sigma') = (\sum\limits_{\sigma \in \Sigma} weight(\sigma)) - 1$ and, since $main(\Sigma') = main(\Sigma)$ and $\mathsf{BV}(\Sigma') = \mathsf{BV}(\Sigma)$, $weight(\Sigma') < weight(\Sigma)$.

$\square$

4.3 Disjunction under Universal Quantification

As mentioned in section 4.1, our procedure to bring a feature term into disjunctive normal form cannot eliminate disjunction under the scope of a universal quantifier, and we haven't provided a way to express such a feature term by means of a set of feature terms. To make the rewrite rules work for those terms, too, we have to extend the system as follows: First, we have to define and prove the correctness of a translation of a feature term with disjunction under the scope of a universal quantifier, then we need to define how an interpretation and a variable assignment satisfy the new constraint, and, after having extended the proof that every feature clause in normal form is satisfiable, we have to enrich the rewrite rules and show again termination and correctness of the new rules. At first, add the following to definition 4.5:

Definition 4.21 *If n is a natural number, r is a set-valued attribute, x is a small variable, $\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n$ and $\boldsymbol{X}$ are big variables, then*

- $x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ *and*
- $\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$

are feature constraints.

The translation of a feature term of the form $\forall r : (\phi_1 \vee ... \vee \phi_n)$ is defined as follows:

Definition 4.22 *If n is a natural number, r is a set-valued attribute, $\phi_1, ..., \phi_n$ are feature terms, x is a small variable, $\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n$ and $\boldsymbol{X}$ are big variables, then*

- $\mathbf{trans}(x, \forall r : (\phi_1 \vee ... \vee \phi_n)) =$
  $\{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\} \cup \mathbf{trans}(\boldsymbol{Y}_1, \phi_1) \cup ... \cup \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$
- $\mathbf{trans}(\boldsymbol{X}, \forall r : (\phi_1 \vee ... \vee \phi_n)) =$
  $\{\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\} \cup \mathbf{trans}(\boldsymbol{Y}_1, \phi_1) \cup ... \cup \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$

Now define satisfiability of the corresponding feature constraints:

Definition 4.23 *For each interpretation $\mathcal{I}$, for each variable assignment α, for each small variable x, for each big variable $\boldsymbol{X}$, $\boldsymbol{Y}_1$, ..., $\boldsymbol{Y}_n$, for each set-valued attribute r,*

- $\mathcal{I}, \alpha \models x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ *iff for each* $u \in \mathcal{D}(r)(\alpha(x))$,
  $u \in \alpha(\boldsymbol{Y}_1) \vee ... \vee u \in \alpha(\boldsymbol{Y}_n)$.
- $\mathcal{I}, \alpha \models \boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ *iff for each* $u' \in \alpha(\boldsymbol{X})$,
  for each $u \in \mathcal{D}(r)(u'), u \in \alpha(\boldsymbol{Y}_1) \vee ... \vee u \in \alpha(\boldsymbol{Y}_n)$.

Then we need to show that the translation function still preserves the denotation of a feature term. As in proposition 4.3, we have to prove two directions. Those parts of the proof which have already been shown are omitted.

Proposition 4.10 *Let $D = D_1 \vee ... \vee D_n$ be in DNF.*
For each D_i, $1 \leq i \leq n$, for each interpretation $\mathcal{I} = \langle \mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{D} \rangle$,
if $x \notin \mathsf{FV}(D)$, then
$[\![D_i]\!]^{\mathcal{I}} = \{\alpha(x) \mid \mathcal{I}, \alpha \models \mathbf{trans}(x, D_i)\}$

Proof. $\supseteq$: If $\mathcal{I}, \alpha \models \mathbf{trans}(x, D_i)$, then $\alpha(x) \in [\![D_i]\!]^{\mathcal{I}}$, if $\mathcal{I}, \alpha \models \mathbf{trans}(\boldsymbol{X}, D_i)$, then $\alpha(\boldsymbol{X}) \subseteq [\![D_i]\!]^{\mathcal{I}}$. Proof by induction on the definition of **trans**.

- $\mathcal{I}, \alpha \models \mathbf{trans}(x, \forall r : (\phi_1 \vee ... \vee \phi_n))$
  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid \forall r : \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}\} \cup \mathbf{trans}(\boldsymbol{Y}_1, \phi_1) \cup ... \cup \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$
  $\Rightarrow$ for each $u \in \mathcal{D}(r)(\alpha(\boldsymbol{X})), u \in \alpha(\boldsymbol{Y}_1)$ or ... or $u \in \alpha(\boldsymbol{Y}_n)$
  and, by induction, $\alpha(\boldsymbol{Y}_1) \subseteq [\![\phi_1]\!]^{\mathcal{I}}_{\alpha}$ and ... and $\alpha(\boldsymbol{Y}_n) \subseteq [\![\phi_n]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow$ for each $u \in \mathcal{D}(r)(\alpha(\boldsymbol{X})), u \in [\![\phi_1]\!]^{\mathcal{I}}_{\alpha}$ or ... or $u \in [\![\phi_n]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(x) \in [\![\forall r : (\phi_1 \vee ... \vee \phi_n)]\!]^{\mathcal{I}}_{\alpha}$
- $\mathcal{I}, \alpha \models \mathbf{trans}(\boldsymbol{X}, \forall r : (\phi_1 \vee ... \vee \phi_n))$
  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}\}\} \cup \mathbf{trans}(\boldsymbol{Y}_1, \phi_1)$
  $\cup ... \cup \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$
  $\Rightarrow$ for each $u' \in \alpha(\boldsymbol{X})$,
  for each $u \in \mathcal{D}(r)(u'), u \in \alpha(\boldsymbol{Y}_1) \vee ... \vee u \in \alpha(\boldsymbol{Y}_n)$ and,
  by induction, $\alpha(\boldsymbol{Y}_1) \subseteq [\![\phi_1]\!]^{\mathcal{I}}_{\alpha}$ and ... and $\alpha(\boldsymbol{Y}_n) \subseteq [\![\phi_n]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow$ for each $u' \in \alpha(\boldsymbol{X}), u' \in [\![\forall r : (\phi_1 \vee ... \vee \phi_n)]\!]^{\mathcal{I}}_{\alpha}$
  $\Rightarrow \alpha(\boldsymbol{X}) \subseteq [\![\forall r : (\phi_1 \vee ... \vee \phi_n)]\!]^{\mathcal{I}}_{\alpha}$

$\subseteq$: If $u \in [\![D_i]\!]^{\mathcal{I}}_{\alpha}$, where $\alpha(x) = u$, then there is an α' s.t. $\alpha'(x) = u$, for each $y \in \mathsf{FV}(D), \alpha'(y) = \alpha(y)$ and $\mathcal{I}, \alpha' \models \mathbf{trans}(x, D_i)$. If $\mathcal{U}' \subseteq [\![D_i]\!]^{\mathcal{I}}_{\alpha}$, where $\alpha(\boldsymbol{X}) = \mathcal{U}'$, then there is an α' s.t. $\alpha'(\boldsymbol{X}) = \mathcal{U}'$, for each $y \in \mathsf{FV}(D), \alpha'(y) = \alpha(y)$ and $\mathcal{I}, \alpha' \models \mathbf{trans}(\boldsymbol{X}, D_i)$.

- $u \in [\![\forall r : (\phi_1 \vee ... \vee \phi_n)]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(x) = u$ and
  $\mathbf{trans}(x, \forall r : (\phi_1 \vee ... \vee \phi_n)) = \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\}$
  $\cup\, \mathbf{trans}(\boldsymbol{Y}_1, \phi_1) \cup ... \cup \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$
  $\Rightarrow \exists \beta. \forall u' \in \mathcal{D}(r)(u), u' \in \beta(\boldsymbol{Y}_1)$ or ... or $u' \in \beta(\boldsymbol{Y}_n)$,
  for each $z \in \mathsf{FV}(D)$, $\beta(z) = \alpha(z)$
  and $\mathcal{I}, \beta \models \mathbf{trans}(\boldsymbol{Y}_1, \phi_1), ..., \mathcal{I}, \beta \models \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$
  (by induction, since $\boldsymbol{Y}_1 \notin \mathsf{FV}(\phi_1)$, ..., $\boldsymbol{Y}_n \notin \mathsf{FV}(\phi_n)$)
  $\Rightarrow \mathcal{I}, \beta_{[x \mapsto u]} \models \{x \mid \forall r : \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}\} \cup \mathbf{trans}(\boldsymbol{Y}_1, \phi_1)$
  $\cup ... \cup \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$
  (since $x \notin \mathsf{FV}(\mathbf{trans}(\boldsymbol{Y}_1, \phi_1)), ..., x \notin \mathsf{FV}(\mathbf{trans}(\boldsymbol{Y}_n, \phi_n)))$
  $\Rightarrow \mathcal{I}, \beta_{[x \mapsto u]} \models \mathbf{trans}(x, \forall r : (\phi_1 \vee ... \vee \phi_n))$

- $\mathcal{U}' \subseteq [\![\forall r : T]\!]^{\mathcal{I}}_{\alpha}$ where $\alpha(\boldsymbol{X}) = \mathcal{U}'$ and $\mathbf{trans}(\boldsymbol{X}, \forall r : (\phi_1, ..., \phi_n))$
  $= \{\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\} \cup \mathbf{trans}(\boldsymbol{Y}_1, \phi_1) \cup ... \cup \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$
  $\Rightarrow \exists \beta.$ for each $u' \in \alpha(\boldsymbol{X})$,
  for each $u \in \mathcal{D}(r)(u'), u \in \beta(\boldsymbol{Y}_1)$ or ... or $u \in \beta(\boldsymbol{Y}_n)$,
  for each $z \in \mathsf{FV}(D)$, $\beta(z) = \alpha(z)$
  and $\mathcal{I}, \beta \models \mathbf{trans}(\boldsymbol{Y}_1, \phi_1), ..., \mathcal{I}, \beta \models \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$
  (by induction, since $\boldsymbol{Y}_1 \notin \mathsf{FV}(\phi_1)$, ... , $\boldsymbol{Y}_n \notin \mathsf{FV}(\phi_n)$)
  $\Rightarrow \mathcal{I}, \beta_{[\boldsymbol{X} \mapsto \mathcal{U}']} \models \{\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}\}$
  $\cup\ \mathbf{trans}(\boldsymbol{Y}_1, \phi_1) \cup ... \cup \mathbf{trans}(\boldsymbol{Y}_n, \phi_n)$
  (since $\boldsymbol{X} \notin \mathsf{FV}(\mathbf{trans}(\boldsymbol{Y}_1, \phi_1)), ..., \boldsymbol{X} \notin \mathsf{FV}(\mathbf{trans}(\boldsymbol{Y}_n, \phi_n)))$
  $\Rightarrow \mathcal{I}, \beta_{[\boldsymbol{X} \mapsto \mathcal{U}']} \models \mathbf{trans}(\boldsymbol{X}, \forall r : (\phi_1 \vee ... \vee \phi_n))$

$\square$

We will need four additional normal form conditions and hence four more rewrite rules.[14]

Definition 4.24 (Additional Normal Form Conditions) *For each small variable x, y, z, for each big variable $\boldsymbol{X}$, $\boldsymbol{Y}_1$, ... $\boldsymbol{Y}_n$, for each $r \in \mathcal{R}$,*

15. *If $x \mid \forall r : \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}, x \mid r : z \in \Sigma$, then for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_i - z \in \Sigma$.*
16. *If $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$, $\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}, \boldsymbol{X} \mid r : z \in \Sigma$, then for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_i - z \in \Sigma$.*
17. *If $\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}, z \mid r : z', \boldsymbol{X} - z \in \Sigma$, then for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_i - z' \in \Sigma$.*
18. *If $\boldsymbol{X} - x, \boldsymbol{X} \mid r : z, x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\} \in \Sigma$, then for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_i - z \in \Sigma$.*

The additional normal form conditions correspond to conditions $10a$, $10b$, 11 and 14 in definition 4.13. Since the set $\{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ arises from a disjunction, it should be evident that in the new conditions we only need *some* $\boldsymbol{Y}_i$ s.t. $\boldsymbol{Y}_i - z \in \Sigma$.

We will see that feature clauses in normal form remain satisfiable:

Proposition 4.11 *For each feature clause Σ, if Σ is in normal form, then Σ is satisfiable.*

[14] An alternative but not very different approach would be to consider rules 15, 16, 17 and 18 as special cases of rules $10a$, $10b$, 11 and 14, where there is only one disjunct under the scope of a universally quantified set-valued attribute r.

We prove again that Σ', which is constructed from a set of constraints Σ in normal form as in definition 4.15, is satisfiable, since the satisfiability of Σ follows from the satisfiability of Σ'.

Let $\mathcal{I}_T$ be the interpretation constructed in definition 4.15 and let the variable assignment be as in the proof of proposition 4.6. Now, it suffices to show the following for feature constraints of the form
$x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ and $\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$:

For each $\sigma \in \Sigma'$, $\mathcal{I}, \alpha \models \sigma$ and hence, $\mathcal{I}, \alpha \models \Sigma$.

Proof.

- $x \mid \forall r : \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\} \in \Sigma'$
  $\Rightarrow \alpha(x) = x$
  $\Rightarrow$ for each $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\alpha(\boldsymbol{Y}_i) = \{y \mid \boldsymbol{Y}_i - y \in \Sigma'\}$
  or $\alpha(\boldsymbol{Y}_i) = \{y \mid \boldsymbol{Y}_i - y \in \Sigma'\} \cup \{u_{\mathbf{y}_i}\}$
  - $y \in \mathcal{D}(r)(x)$ s.t. $x \mid r : y \in \Sigma'$
    $\Rightarrow$ for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_i - y \in \Sigma'$
    $\Rightarrow y \in \alpha(\boldsymbol{Y}_i)$ (by cond. 15)
  - $y \in \mathcal{D}(r)(x)$ s.t. for some $\boldsymbol{X}$, $\boldsymbol{X} - x$ and $\boldsymbol{X} \mid r : y \in \Sigma'$
    $\Rightarrow$ for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_i - y \in \Sigma'$ (by cond. 18)
    $\Rightarrow y \in \alpha(\boldsymbol{Y}_i)$

  $\Rightarrow$ for each $u \in \mathcal{D}(r)(\alpha(x))$, $u \in \alpha(\boldsymbol{Y}_1) \vee ... \vee u \in \alpha(\boldsymbol{Y}_n)$

- $\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\} \in \Sigma'$
  $\Rightarrow \alpha(\boldsymbol{X}) = \{x \mid \boldsymbol{X} - x \in \Sigma'\} \cup \{u_{\mathbf{x}}\}$
  or $\alpha(\boldsymbol{X}) = \{x \mid \boldsymbol{X} - x \in \Sigma'\}$
  $\Rightarrow$ for each $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\alpha(\boldsymbol{Y}_i) = \{y \mid \boldsymbol{Y}_i - y \in \Sigma'\}$
  or $\alpha(\boldsymbol{Y}_i) = \{y \mid \boldsymbol{Y}_i - y \in \Sigma'\} \cup \{u_{\mathbf{y}_i}\}$
  $\Rightarrow$ for each $u' \in \alpha(\boldsymbol{X})$
  - $u' = u_{\mathbf{x}}$
    $\Rightarrow \mathcal{D}(r)(u_{\mathbf{x}}) = \{y \mid \boldsymbol{X} \mid r : y \in \Sigma'\} \subseteq \{y \mid$ for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\boldsymbol{Y} - y_i \in \Sigma'\}$ (by cond. 16)
    $\Rightarrow$ for each $u \in \mathcal{D}(r)(u_{\mathbf{x}}), u \in \alpha(\boldsymbol{Y}_1) \vee ... \vee u \in \alpha(\boldsymbol{Y}_n)$.
  - $u' = x$
    $\Rightarrow \mathcal{D}(r)(x) = \{y \mid x \mid r : y \in \Sigma'\} \cup \{z \mid \boldsymbol{X} \mid r : z \in \Sigma'\}$
    $\subseteq \{y \mid$ for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ... \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_i - y \in \Sigma'\}$
    (by cond. 16 and 17)
    $\Rightarrow$ for each $u \in \mathcal{D}(r)(x), u \in \alpha(\boldsymbol{Y}_1) \vee ... \vee u \in \alpha(\boldsymbol{Y}_n)$.

  $\Rightarrow$ for each $u' \in \alpha(\boldsymbol{X})$,
  for each $u \in \mathcal{D}(r)(u'), u \in \alpha(\boldsymbol{Y}_1) \vee ... \vee u \in \alpha(\boldsymbol{Y}_n)$.

$\square$

Definition 4.25 (Additional Rewrite Rules) *Let Γ be a set of sets of constraints, let Σ be a set of constraints. Then the additional rewrite rules are defined as follows:*

For for each set-valued attribute r,

15. $\Gamma \uplus \{\Sigma \cup \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ x \mid r : z\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ x \mid r : z\ ,\ \boldsymbol{Y}_i - z\} \mid$
$\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$*, and for no* $\boldsymbol{Y}_j \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_j - z \in \Sigma\}$
if there is no $\boldsymbol{Y}_k \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ *s.t.* $\boldsymbol{Y}_k - z \in \Sigma$.

16. $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ \boldsymbol{X} \mid r : z\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ \boldsymbol{X} \mid r : z\ ,\ \boldsymbol{Y}_i - z\} \mid$
$\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$*, and for no* $\boldsymbol{Y}_j \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_j - z \in \Sigma\}$
if $\boldsymbol{X} \in \mathsf{CV}(\Sigma)$ *and there is no* $\boldsymbol{Y}_k \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ *s.t.* $\boldsymbol{Y}_k - z \in \Sigma$.

17. $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ z \mid r : z'\ ,\ \boldsymbol{X} - z\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ z \mid r : z'\ ,\ \boldsymbol{X} - z\ ,\ \boldsymbol{Y}_i - z'\} \mid$
$\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$*, and for no* $\boldsymbol{Y}_j \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_j - z' \in \Sigma\}$
if there is no $\boldsymbol{Y}_k \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ *s.t.* $\boldsymbol{Y}_k - z' \in \Sigma$.

18. $\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - x\ ,\ \boldsymbol{X} \mid r : z\ ,\ x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\}\} \longrightarrow$
$\Gamma \uplus \{\Sigma \cup \{\boldsymbol{X} - x\ ,\ \boldsymbol{X} \mid r : z\ ,\ x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ \boldsymbol{Y}_i - z\} \mid$
$\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$*, and for no* $\boldsymbol{Y}_j \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$, $\boldsymbol{Y}_j - z \in \Sigma\}$
if there is no $\boldsymbol{Y}_k \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ *s.t.* $\boldsymbol{Y}_k - z \in \Sigma$.

In contrast to rules $10a$, $10b$, 11 and 14, the new rewrite rules produce further sets of sets. For each dependency between a set of big variables $\{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ and a small variable z, there is exactly *one* constraint $\boldsymbol{Y}_i - z$ added since only *one* constraint $\boldsymbol{Y}_i - z$ has to be satisfied. This can be seen in the normal form conditions, too. But it is also ensured that, if we have more dependencies, e.g., $\{\Sigma \cup \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}, x \mid r : z,$ $x \mid r : z'\}$, that the rewrite system produces all possible sets $\Sigma \cup \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\},\ x \mid r : z, x \mid r : z',\ \boldsymbol{Y}_i - z\ ,\ \boldsymbol{Y}_j - z'\}$, where $\boldsymbol{Y}_i, \boldsymbol{Y}_j \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$. Then all unsatisfiable sets are removed by application of the rewrite rules, and only the satisfiable sets remain.

The additional condition "if there is no $\boldsymbol{Y}_k \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ s.t. $\boldsymbol{Y}_k - z \in \Sigma$" might seem unnecessary. But since all rewrite rules produce sets of sets, we have to ensure that, provided a rule does not apply, we do not obtain the empty set, which would display unsatisfiability. Now we have to prove the correctness of the additional rewrite rules.

Proposition 4.12 *For each matrix Γ, for each matrix Γ', for each interpretation $\langle \mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{D} \rangle$, for each variable assignment α, if $\Gamma \longrightarrow \Gamma'$, then $\mathcal{I}, \alpha \models \Gamma$ iff $\mathcal{I}, \alpha \models \Gamma'$.*

Proof. Rules 1 to 14 haven't changed, so we only have to consider the correctness of rules 15 to 18:

- For rule 15, we need to show that
  $\mathcal{I}, \alpha \models \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ x \mid r : z\}$

  $\Leftrightarrow$ For some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$,
  $\mathcal{I}, \alpha \models \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ x \mid r : z\ ,\ \boldsymbol{Y}_i - z\}$

  Right to left is obvious:

  $\mathcal{I}, \alpha \models \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ x \mid r : z\ ,\ \boldsymbol{Y}_i - z\}$
  for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$

  $\Rightarrow \mathcal{I}, \alpha \models \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ x \mid r : z\}$

  From left to right we have:

  $\mathcal{I}, \alpha \models \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ x \mid r : z\}$

  $\Rightarrow$ for each $u \in \mathcal{D}(r)(\alpha(x)), u \in \alpha(\boldsymbol{Y}_1)$ or ... or $u \in \alpha(\boldsymbol{Y}_n)$ and $\alpha(z) \in \mathcal{D}(r)(\alpha(x))$

  $\Rightarrow$ For each $u \in \mathcal{D}(r)(\alpha(x)), u \in \alpha(\boldsymbol{Y}_1)$ or ... or $u \in \alpha(\boldsymbol{Y}_n)$ and $\alpha(z) \in \mathcal{D}(r)(\alpha(x))$ and there is a $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$ s.t. $\alpha(z) \in \alpha(\boldsymbol{Y}_i)$

  $\Rightarrow$ for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$,
  $\mathcal{I}, \alpha \models \{x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ x \mid r : z\ ,\ \boldsymbol{Y}_i - z\}$

- For rules 16, 17 we have to prove two directions, too. As in the correctness proof of rule 15, the step from right to left is obvious, since there is always a constraint of the form $\boldsymbol{Y}_i - z$ added, the only difference being that we consider *some* big variable $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$. The proofs from left to right closely resemble the proofs of rules $10b$ and 11 in proposition 4.8 and are therefore omitted.

- Though the proof that rule 18 is correct, is very similar to the proof of rule 14, it might not seem too obvious:
  For rule 18, we need to show that
  $\mathcal{I}, \alpha \models \{\boldsymbol{X} - x\ ,\ \boldsymbol{X} \mid r : z\ ,\ x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\}$

  $\Leftrightarrow$ For some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$,
  $\mathcal{I}, \alpha \models \{\boldsymbol{X} - x\ ,\boldsymbol{X} \mid r : z\ ,\ x \mid \forall r : \{\boldsymbol{Y}_1, .., \boldsymbol{Y}_n\}\ ,\ \boldsymbol{Y}_i - z\}$

  Right to left is obvious:

  $\mathcal{I}, \alpha \models \{\boldsymbol{X} - x,\ \boldsymbol{X} \mid r : z\ ,\ x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ \boldsymbol{Y}_i - z\}$
  for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$

  $\Rightarrow \mathcal{I}, \alpha \models \{\boldsymbol{X} - x,\ \boldsymbol{X} \mid r : z,\ x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\}$

From left to right we have:

$\mathcal{I}, \alpha \models \{\boldsymbol{X} - x\ ,\ \boldsymbol{X} \mid r : z,\ x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\}$

$\Rightarrow \alpha(x) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X})$, $\alpha(z) \in \mathcal{D}(r)(u)$, for each $u' \in \mathcal{D}(r)(\alpha(x))$, $u' \in \alpha(\boldsymbol{Y}_1)$ or ... or $u' \in \alpha(\boldsymbol{Y}_n)$

$\Rightarrow \alpha(x) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X})$, $\alpha(z) \in \mathcal{D}(r)(u)$, for each $u' \in \mathcal{D}(r)(\alpha(x))$, $u' \in \alpha(\boldsymbol{Y}_1)$ or ... or $u' \in \alpha(\boldsymbol{Y}_n)$ and $\alpha(z) \in \mathcal{D}(r)(\alpha(x))$

$\Rightarrow \alpha(x) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X})$, $\alpha(z) \in \mathcal{D}(r)(u)$, for each $u' \in \mathcal{D}(r)(\alpha(x))$, $u' \in \alpha(\boldsymbol{Y}_1)$ or ... or $u' \in \alpha(\boldsymbol{Y}_n)$, $\alpha(z) \in \mathcal{D}(r)(\alpha(x))$ and $\alpha(z) \in \alpha(\boldsymbol{Y}_1)$ or ... or $\alpha(z) \in \alpha(\boldsymbol{Y}_n)$

$\Rightarrow \alpha(x) \in \alpha(\boldsymbol{X})$, for each $u \in \alpha(\boldsymbol{X})$, $\alpha(z) \in \mathcal{D}(r)(u)$, for each $u' \in \mathcal{D}(r)(\alpha(x))$, $u' \in \alpha(\boldsymbol{Y}_1)$ or ... or $u' \in \alpha(\boldsymbol{Y}_n)$, $\alpha(z) \in \mathcal{D}(r)(\alpha(x))$ and for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$, $\alpha(z) \in \alpha(\boldsymbol{Y}_i)$

$\Rightarrow$ for some $\boldsymbol{Y}_i \in \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}$,
$\mathcal{I}, \alpha \models \{\boldsymbol{X} - x,\ \boldsymbol{X} \mid r : z,\ x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_n\}\ ,\ \boldsymbol{Y}_i - z\}$

$\square$

The only thing that remains to be shown is the termination of the additional rewrite rules: Since we require the number of disjuncts to be finite, all of the new rules produce a finitely branching tree. Hence, each set of constraints which is produced by such a rewrite rule decreases the weight of a feature clause by 1.

Note that the termination of rule $1b$ is affected by the new feature constraints: Its application does not terminate after $2 \times (|\Sigma_1| \times ... \times |\Sigma_k|)$ steps. Our new feature constraints can contain more than two (big or small) variables, but not more than finitely many. So, rule $1b$ terminates after at most $(n+1) \times (|\Sigma_1| \times ... \times |\Sigma_k|)$ steps, where n is the maximum of all n' of all feature constraints s.t. $x \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_{n'}\}$,
$\boldsymbol{X} \mid \forall r : \{\boldsymbol{Y}_1, ..., \boldsymbol{Y}_{n'}\} \in \Sigma$.

Again, as in the proof of proposition 4.9, the weight of the feature clauses can never become negative.

4.4 Open Questions

4.4.1 Path Equations

In the our extended feature logic, we are not able to express path equations. If we have, for example, a feature term $\forall r : (f : x \wedge g : x)$, where $x \in \mathsf{VAR}$, f and $g \in \mathcal{F}$, then that does not only mean that for each r, the value of f equals the value of g, but that each value of f under r equals each value of g under r. In our logic, we only have variables to express equality. If we add path equations as additional feature terms to our feature logic, it is not clear how we could translate such a term into a set of feature constraints. Presumably, one would need a further kind of feature constraint and, consequently, even more normal form conditions.

4.4.2 Cycles

Manandhar (1993) proves that if a set of constraints (in a constraint language which is comparable to ours) contains *cycles*, i.e., for some variable x there is a non-empty path which leads from x to x,[15] *and* the constraint language allows to express path equations, then constraint solving becomes undecidable. He therefore imposes an *acyclicity condition* to exclude cycles.

Since we proved decidability without banning cycles, they apparently present no problem for our constraint solving procedure. I conjecture, though, that they would do so, if we included path equations in our logic.

5 Towards a Treatment of LP Rules

Following (Manandhar 1995b), I will outline how our feature logic with set-valued attributes could be used to describe linear precedence (LP) relations and which additional definitions would be needed.

5.1 LP Rules

LP rules are word order constraints. They are expressed by an antisymmetric transitive relation, usually written $<$. A constraint $X < Y$ states that a daughter with property X precedes all of its sisters with property Y.[16] Such statements have to be true for all phrases in a language. With these constraints we are able to leave the word order of a phrase underspecified. If we know, for example, that $A < B$ and $A < C$, we don't say anything about the relation between B and C.

Pollard and Sag (1987) introduce LP rules for HPSG, but leave open the question of how LP constraints could be integrated in the formal system, i.e., how the phonology of a phrase is related to the phonology of its daughters.

5.2 Further Feature Terms

Our (extended) feature logic offers the possibility to describe relations. In order to express LP constraints, one would also need the transitive

[15] For example, the set $\Sigma = \{x \mid f : y,\, y \mid x\}$ contains a cycle.

[16] See (Pollard and Sag 1987) and also (Gazdar, Klein, Pullum, and Sag 1985).

(or the reflexive-transitive) closure of set-valued attributes: With this addition, we would be able to frame the idea of two entities being related by the transitive closure of a set-valued attribute, without mentioning the "length" of this relation.

Manandhar (1995b) enriches an untyped feature logic with set-valued attributes and several feature constraints like the *transitive* and *reflexive-transitive closure*, *domain precedence*, *first daughter* and *subset inclusion* to describe LP relations.

We will start by defining the transitive closure and then show how domain precedence, which is a possibility to capture precedence relations between word-ordering domains, could be treated in our feature logic.

5.2.1 Transitive Closure

In the following, I will present a brief sketch of how the transitive closure of set-valued attributes could be treated in our approach.[17]

Definition 5.1 (Transitive Closure) *Let p be a relation over a set $\mathcal{U}$. Then the* transitive closure p^+ *of p is defined as follows:*
For each $u, u', u'' \in \mathcal{U}$, $\langle u, u'' \rangle \in p^+$ iff $\langle u, u'' \rangle \in p \vee (\exists u' : \langle u, u' \rangle \in p \wedge \langle u', u'' \rangle \in p^+)$.

Definition 5.2 (Feature Terms with Transitive Closure) *Add the following feature terms to definition 2.2:*

- $r^+ : \{\phi_1, ..., \phi_n\}$ *is a term if* $r \in \mathcal{R}$, $n \geq 1$ *and* $\phi_1, ..., \phi_n$ *are terms.*
- $\forall r^+ : \phi$ *is a term* *if* $r \in \mathcal{R}$ *and* ϕ *is a term.*

Furthermore, the interpretation of the transitive closure of set-valued features is defined inductively as follows:

- For each $r \in \mathcal{R}$, for each $u \in \mathcal{U}$,
  $\mathcal{D}(r^+)(u) = \{u' \in \mathcal{U} \mid u' \in \mathcal{D}(r)(u)\} \cup$
  $\{u'' \mid \exists u' : u' \in \mathcal{D}(r)(u) \wedge u'' \in \mathcal{D}(r^+)(u')\}$.

Definition 5.3 (Term Interpretation with Transitive Closure) *Add the following denotations to definition 3.4:*

- $[\![r^+ : \{\phi_1, ...\phi_n\}]\!]^{\mathcal{I}}_{\alpha} = \{u \in \mathcal{U} \mid \exists u_1, ...u_n : u_1, ...u_n \in \mathcal{D}(r^+)(u) \wedge u_1 \in [\![\phi_1]\!]^{\mathcal{I}}_{\alpha} \wedge ... \wedge u_n \in [\![\phi_n]\!]^{\mathcal{I}}_{\alpha}\}$
- $[\![\forall r^+ : \phi]\!]^{\mathcal{I}}_{\alpha} = \{u \in \mathcal{U} \mid \forall u' : u' \in \mathcal{D}(r^+)(u) \rightarrow u' \in [\![\phi]\!]^{\mathcal{I}}_{\alpha}\}$

[17] The reflexive-transitive closure p^* of a relation p over a set $\mathcal{U}$ can easily be defined in terms of the transitive closure: $p^* = p^+ \cup \{\langle u, u \rangle, u \in \mathcal{U}\}$.

Clearly, one can apply the equations $\neg(r : \phi) \equiv \forall r : \neg\phi$ and $\neg(\forall r : \phi) \equiv r : \neg\phi$ to the transitive closure r^+ of r, too. The proof is analogous to the proofs of proposition 4.1 and is omitted here.

Definition 5.4 (trans) *Let*

- $\mathbf{trans}(x, r^+ : T) = \{x \mid r^+ : y\} \cup \mathbf{trans}(y, T)$,
- $\mathbf{trans}(\boldsymbol{X}, r^+ : T) = \{\boldsymbol{X} \mid r^+ : y\} \cup \mathbf{trans}(y, T)$,
- $\mathbf{trans}(x, \forall r^+ : T) = \{x \mid \forall r^+ : \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, T)$, *and*
- $\mathbf{trans}(\boldsymbol{X}, \forall r^+ : T) = \{\boldsymbol{X} \mid \forall r^+ : \boldsymbol{Y}\} \cup \mathbf{trans}(\boldsymbol{Y}, T)$.

Then the proof that the translation preserves the denotation works as in proposition 4.3.

If we consider the normal form in definition 4.13, we will need additional conditions: 8 and 9 should hold for the transitive closure of r as well. Conditions 10, 11 and 14 should be true for either combination of r and r^+ in $x \mid \forall r^{(+)} : \boldsymbol{Y}$ and $x \mid r^{(+)} : z$.

To ensure that the normal form rules work for feature terms containing transitive closure of set-valued attributes, we will need additional rules like "If $x \mid r^{(+)} : y$ and $y \mid r^{(+)} : z$ then $x \mid r^+ : z$." for small variables and constraint variables.[18] If we leave the interpretation in definition 4.15 fixed, we will see that the feature constraints $x \mid r^+ : y$, $\boldsymbol{X} \mid r^+ : y$, $x \mid \forall r^+ : \boldsymbol{Y}$ and $\boldsymbol{X} \mid \forall r^+ : \boldsymbol{Y}$, provided they are members of a feature clause in normal form, are satisfiable, too.

5.2.2 Domain Precedence

Manandhar (1995b) introduces *domain precedence* as a feature constraint in his logic. Given domain precedence, we can express the idea that each element in a set, called *domain*, is related to each member of another domain via the transitive (or reflexive-transitive) closure of a set-valued attribute. Manandhar (1995b) defines the satisfiability of a feature constraint describing domain precedence as follows:

- $\mathcal{I}, \alpha \models f(x) : p^+ : g(y)$
  $\Leftrightarrow \forall e_1, e_2 \in \mathcal{U}^I((e_1 \in f^I(\alpha(x)) \wedge e_2 \in g^I(\alpha(y))) \Rightarrow (e_1, e_2) \in (p^I)^+)$

Unfortunately, we haven't got the possibility to quantify over pairs of variables. Therefore, if we wanted to express domain precedence in our feature logic, we would again have more equality than we need as in section 4.4.1.

[18] This could perhaps, together with the other additional rules, be handled more efficiently.

6 Conclusion

We have provided a typed feature logic with the possibility to express relations and also a decision procedure to check the satisfiability of feature terms. Moreover, we have outlined how (reflexive-)transitive closure of set-valued attributes could be handled in the logic and in the constraint solving mechanism and which further extensions would be desirable in order to model LP constraints within the logic.

The decision procedure for the extended logic, our rewrite system, is far more complex than a constraint solving mechanism for a typed feature logic without set-valued attributes. It remains unclear how we could include path equations and which consequences the combination of path equations and cycles has for the decidability of the logic.

As a result, we have to reflect upon the following problem: How can we add further feature terms to our logic in order to make a practical application possible without losing decidability? Which restrictions can be imposed on the formal system without losing too much expressive power?

Several interesting questions are still unanswered and could provide a good basis for further research in both linguistics and feature logics.

References

Dörre, J. and M. Dorna (1993, August). CUF - a formalism for linguistic knowledge representation. In J. Dörre (Ed.), *Computational aspects of constraint based linguistic descriptions I*, pp. 1–22. Universität Stuttgart: DYANA-2 Deliverable R1.2.A.

Ebbinghaus, H.-D., J. Flum, and W. Thomas (1992). *Einführung in die mathematische Logik*. Mannheim, Leipzig, Wien, Zürich: B.I.-Wissenschaftsverlag.

Gazdar, G., E. Klein, G. Pullum, and I. Sag (1985). *Generalized Phrase Structure Grammar*. Oxford: Blackwell.

Götz, T. (in prep.). *Feature Constraint Grammars*. Ph. D. thesis, Universitliät Tübingen.

Höhfeld, M. and G. Smolka (1988). Definite relations over constraint languages. Technical Report 53, LILOG, IBM Deutschland.

Jaffar, J. and J.-L. Lassez (1987). Constraint logic programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, Munich, Germany, pp. 111–119.

King, P. J. (1989). *A logical formalism for head-driven phrase structure grammar*. Ph. D. thesis, University of Manchester.

Manandhar, S. (1993). *Relational Extensions to Feature Logic: Application to Constraint Based Grammars*. Ph. D. thesis, Department of Artificial Intelligence, University of Edinburgh.

Manandhar, S. (1995a). An attributive logic of set descriptions and set operations: Extended report. *Edinburgh Working Papers in Cognitive Science Nonclassical Feature Systems*(10), 39–60.

Manandhar, S. (1995b). Deterministic consistency checking of LP constraints. In *proceedings of the 7th conference of the EACL*, Dublin, Ireland, pp. 165–172.

Pollard, C. and I. A. Sag (1987). *Information-Based Syntax and Semantics*. CSLI lecture notes.

Pollard, C. and I. A. Sag (1994). *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Richter, F. and M. Sailer (1995). Remarks on Linearization. Reflections on the Treatment of LP-Rules in HPSG in a Typed Feature Logic. Master's thesis, Universität Tübingen.

Smolka, G. (1988). A feature logic with subsorts. LILOG Report 33, IWBS, IBM Deutschland.