Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT

How to Prove Higher Order Theorems in First Order Logic

Manfred Kerber

SEKI Report SR-90-19 (SFB)

How to Prove Higher Order Theorems in First Order Logic

Manfred Kerber
Fachbereich Informatik, Universität Kaiserslautern
D-6750 Kaiserslautern, West Germany
UUCP: kerber@informatik.uni-kl.de

Abstract

In this paper we are interested in using a first order theorem prover to prove theorems that are formulated in some higher order logic. To this end we present translations of higher order logics into many sorted first order logic with equality and give a sufficient criterion for the soundness of these translations. In addition translations are introduced that are sound and complete with respect to L. Henkin's general model semantics. Our higher order logics are based on a restricted type structure in the sense of A. Church, they have typed function symbols and predicate symbols, but no sorts. The translation results are finally generalized to handle such a logic with equality.

Keywords: higher order logic, second order logic, general model semantics, translation, sorted first order logic, morphism, soundness, completeness

Contents

This work was supported by the Deutsche Forschungsgemeinschaft, SFB 314 (D2,D3)

1 INTRODUCTION

Die Grenzen meiner Spache bedeuten die Grenzen meiner Welt.
Ludwig Wittgenstein,
Tractatus logico-philosophicus 5.6

First order logic is a powerful tool for expressing and proving mathematical facts. Nevertheless higher order expressions are often better suited for the representation of mathematics and in fact almost all mathematical text books rely on some higher order fragments for expressiveness. In order to prove such theorems mechanically there are two options: either to have a theorem prover for higher order logic such as TPS (of P. ANDREWS [2]) or to translate the higher order constructs into corresponding first order expressions and to use a first order theorem prover. As important as the first development is – which may be the way of the future – we follow the second approach because strong first order theorem provers are available today.

THE LIMITATIONS OF FIRST ORDER LOGIC

First order logic and the set theories of ZERMELO-FRAENKEL [17, 7] or VON NEUMANN-GÖDEL-BERNAYS [8] have been developed for the formalization of mathematical concepts and for reasoning about them. Other approaches are RUSSEL'S ramified theory of types [14] and CHURCH'S simple theory of types [5] which formalize higher order logic. Mathematicians use a (compared to the formal approaches) informal technical language that is much closer to higher order logic augmented by "naive" set theory than to first order logic. They know about the antinomies and avoid them, for example by the ommission of expressions like "$\{x|x \notin x\}$". They also know that there is a (hopefully) clean foundation of set theory, how this is done in detail is in general however not of much interest to a working mathematician (if he is not working on the foundations of mathematics like logic or set theory).

Formal set theory is of course a very strong tool, especially when higher concepts are introduced by abbreviations. Beginning with the binary relation "$\in$" one can (and this is really done by N. BOURBAKI [4]) define the concepts subset, intersection, union, function, relation, powerset, and so on. The definition of a function as a left-total, right-unique relation is rather complex and remote from the construct of a function symbol that is provided originally in logic in order to express functions. The representation of concepts using functions is more adequate in a higher order language. For instance in higher order logic it is possible to write:
$\forall + \ \ associative(+) \iff \forall x, y, z \ \ (x + y) + z \equiv x + (y + z)$.
Here $+$ is a function variable, and *associative* is a predicate constant, which expects a function term as its argument. This cannot be written immediately in first order logic, because we quantify over $+$, so $+$ would have to be a variable. On the other hand it must be a function because of the term $x + y$, hence a function variable, and this is excluded in first order logic. Nevertheless this definition is expressable in first order logic. Many concepts cannot be axiomatized in first order logic at all, for example the set $\mathbb{N}$ of natural numbers is not first order characterizable. G. PEANO uses in his axiomatization of the natural numbers the following induction axiom, which is second order:
$\forall P \ \ P(0) \wedge (\forall n^{nat} \ \ P(n) \Longrightarrow P(s(n))) \Longrightarrow (\forall n^{nat} \ \ P(n))$

Another example of the inadequacy of first order logic comes from the theorem of LÖWENHEIM-SKOLEM: Every (countable) axiomatization of a set which has an infinite model also has a countable model. Therefore every first order axiomatization of the real numbers $\mathbb{R}$ has a countable model.

WHY AND HOW TRANSLATION

Representing knowledge in an adequate way – adequate with respect to the naturalness of the representation of the object – is one thing, the other thing is to have an adequate and strong form of reasoning. If one uses higher order logic there are two possibilities: either to build strong higher order

theorem provers or to translate into first order logic. We shall follow the second approach in this paper.

A common translation of our formula above in a first order logic with equality is:

$\forall +\ \ associative(+) \iff \forall x, y, z\ \ apply(+, apply(+, x, y), z) \equiv apply(+, x, apply(+, y, z))$

Here *apply* is a new function constant and + an object variable. Although *apply* is interpreted freely it is intended that the interpretation of $apply(+, x, y)$ is exactly the same as the interpretation of the higher order term $x + y$.

Another translation, which does not use equality is:

$$\forall +\ \ associative(+) \iff \forall x, y, z, u, v, w\ \ apply(+, x, y, u) \wedge apply(+, u, z, w) \wedge apply(+, y, z, v) \Longrightarrow apply(+, x, v, w)$$

Here *apply* is a predicate; again it is interpreted freely, although it is intended that z is the sum of x and y in $apply(+, x, y, z)$. In other words different translations from higher to first order logic are possible.

Hence the following problems:

- Under what conditions can such a translation be correct? That is, if we translate a formula and we obtain a tautology, when is the original formula a tautology too?
- In what sense can such a translation be complete? That is, if we translate a tautology, do we always obtain a tautology?

1.1 Example: Consider the following tautology:
$\forall P, Q\ \ ((\forall x\ \ P(x) \Longrightarrow Q(x)) \wedge \forall x\ \ P(x) \Longrightarrow \forall x\ \ Q(x))$
This can be translated into the first order formula
$\forall P, Q\ \ ((\forall x\ \ apply(P, x) \Longrightarrow apply(Q, x)) \wedge \forall x\ \ apply(P, x) \Longrightarrow \forall x\ \ apply(Q, x))$
This is obviously a tautology again, hence in this case the translation is sound and complete.

1.2 Example: Consider the following tautology with function constants f and g:
$\forall x\ \ f(x) \equiv g(x) \Longrightarrow f \equiv g$
This is a tautology because functions which have the same results for all arguments are equal (extensionality). It is translated to
$\forall x\ \ apply(f, x) \equiv apply(g, x) \Longrightarrow f \equiv g$
But this is not a tautology: by interpreting *apply* as the projection to the second component and f and g as different elements we obtain a counterexample. This translation is obviously not complete.

For general considerations concerning the expressiveness of higher order logic, it is obvious that if we find a translation from higher order to first order logic, it cannot be complete in the general sense, especially since the theorem of LÖWENHEIM-SKOLEM must hold and because of K. GÖDEL'S incompleteness result. In principle such a translation must be equivalent to some set theoretical formulation as stated in A. MOSTOWSKI'S isomorphism theorem [10].*

RELATED WORK

J. VAN BENTHEM and K. DOETS [3] give a translation of a restricted higher order logic without function symbols and without higher order constants and identities to a standard first order logic. They introduced the general idea of a translation, and its soundness and completeness. The translation to standard first order logic leads to more complicated formulae than the translation to a sorted version, because it is necessary to relativize quantification with respect to the corresponding type.

Of great influence for the present paper are the translation techniques of H. J. OHLBACH [12], who translates modal logics and other non-classical logics to a context logic; where contexts are restricted higher order expressions. These contexts are translated to a order sorted first order logic.

Here a translation of (almost) full higher order logic with function symbols to a many sorted first order logic with equality is given. We do not need a general order sorted logic as long as we do not use a sorted higher order source logic. In chapter 7 we give an extension for a higher order logic with equality.

*I like to thank Heinrich Herre for introducing me to this work of Mostowski.

2 Higher Order Logic

In this section we define formally a higher order logic based on Church's simple theory of types, much of the notation is taken from [1]. However, we shall write the types in a different way. For example if a and b are individual constants and P is a binary predicate symbol on individuals, we write its type as $(\iota \times \iota \to o)$ instead of $(o\iota\iota)$ for better readability. Apologies to all who are familiar with Church's original notation.

The Syntax

Let us introduce type symbols first, then define terms and formulae for the logics $\mathcal{L}^\omega$. The n-th order predicate logics $\mathcal{L}^n$ are then defined as subsets of $\mathcal{L}^\omega$.

2.1 Definition (Types of $\mathcal{L}^\omega$):

1. ι is a type of order 0 that denotes the type of the individuals.
2. o is a type of order 1. It denotes the type of the truth values.
3. If $\tau_1, \ldots, \tau_m$, and σ are type symbols not equal to o (with $m \geq 1$) then $(\tau_1 \times \cdots \times \tau_m \to \sigma)$ is a type of order 1 + maximum of the orders of $\tau_1, \ldots, \tau_m, \sigma$. It denotes the type of m-ary functions with arguments of type $\tau_1, \ldots, \tau_m$, respectively, and value of type σ.
4. If $\tau_1, \ldots, \tau_m$ are type symbols not equal to o (with $m \geq 1$) then $(\tau_1 \times \cdots \times \tau_m \to o)$ is a type of order 1 + maximum of the orders of $\tau_1, \ldots, \tau_m$. It denotes the type of m-ary predicates with arguments of type $\tau_1, \ldots, \tau_m$, respectively.

2.2 Remark: We exclude – unlike Church and Andrews [5, 1] – types like $(o \to o)$, because these expressions are not translatable by our method, see remark 6.2. Therefore in our restricted language it is not possible to define the connectives $\neg$ and $\wedge$ and the quantifier $\forall$, and hence they must be introduced as primitives. Nevertheless we assume that the languages $\mathcal{L}^n$ – defined below – are adequate for expressing most mathematical facts. For instance we can have predicates like $ordered\_group(G, +, \leq)$ of type $(\iota \times (\iota \times \iota \to \iota) \times (\iota \times \iota \to o) \to o)$*.

2.3 Definition (Signature of $\mathcal{L}^\omega$): The signature of a logic in $\mathcal{L}^\omega$ is a set $\mathcal{S} = \bigcup_\tau \mathcal{S}_\tau^{const} \cup \bigcup_\tau \mathcal{S}_\tau^{var}$ where each set $\mathcal{S}_\tau^{const}$ is a (possibly empty) set of constant symbols of type τ and $\mathcal{S}_\tau^{var}$ a countable infinite set of variable symbols of type τ. We assume that the sets $\mathcal{S}_\tau$ are all disjoint, in addition we sometimes mark the elements of a set $\mathcal{S}_\tau$ by its type τ as index. A logic in $\mathcal{L}^\omega$ is defined by its signature $\mathcal{S}$ and is denoted $\mathcal{L}^\omega(\mathcal{S})$. If there is only one signature and no danger of confusion we also write $\mathcal{L}^\omega$ instead of $\mathcal{L}^\omega(\mathcal{S})$.

2.4 Definition (Terms of $\mathcal{L}^\omega$):

1. Every variable or constant of a type τ is a term.
2. If $f_{(\tau_1 \times \cdots \times \tau_m \to \sigma)}, t_{\tau_1}, \ldots, t_{\tau_m}$ are terms of the type indicated by their subscripts, then $f_{(\tau_1 \times \cdots \times \tau_m \to \sigma)}(t_{\tau_1}, \ldots, t_{\tau_m})$ is a term of type σ.

2.5 Definition (Formulae of $\mathcal{L}^\omega$):

1. Every term of type o is a formula.
2. If φ and ψ are formulae and x is a variable of any type, then $(\neg\varphi)$, $(\varphi \wedge \psi)$, and $(\forall x \varphi)$ are formulae. As long as there is no danger of confusion we sometimes omit parentheses.

*It is also possible to represent the set G as object of type $(\iota \to o)$.

2.6 Definition (Formulae of $\mathcal{L}^{\omega}_{\equiv}$):

1. Every term of type o is a formula.
2. If t_1 and t_2 are terms of type τ with $\tau \neq o$ then $(t_1 \equiv_{(\tau\times\tau\to o)} t_2)$ is a formula.
3. If φ and ψ are formulae and x is a variable of any type, then $(\neg\varphi)$, $(\varphi \wedge \psi)$, and $(\forall x\varphi)$ are formulae.

Of course we have to add $\equiv_{(\tau\times\tau\to o)}$ to $\mathcal{S}^{const}_{(\tau\times\tau\to o)}$. We use the "$\equiv$" as elements of the signature and for the equalities in sets. They have the usual semantics. "$=$" is used as equality symbol at the meta-level.

2.7 Remark: As usual one can define $\vee$, $\Longrightarrow$, $\Longleftrightarrow$, and $\exists$ in terms of $\neg$, $\wedge$, and $\forall$ and use formulae containing these symbols as abbreviations.

2.8 Definition ($\mathcal{L}^n$, for $n \geq 1$): $\mathcal{L}^{2n}$ ($\mathcal{L}^{2n}_{\equiv}$) is a subset of $\mathcal{L}^{\omega}$ ($\mathcal{L}^{\omega}_{\equiv}$) so that every variable and every constant is of order less or equal to n, $\mathcal{L}^{2n-1}$ ($\mathcal{L}^{2n-1}_{\equiv}$) is a subset of $\mathcal{L}^{2n}$ ($\mathcal{L}^{2n}_{\equiv}$) such that no variable of order n is quantified.

The Semantics

The standard semantics is due to A. Tarski and has been extended by L. Henkin [9] to the general model semantics, we shall follow these concepts.

We use the following notation: Let $A_1, \ldots, A_m$, and B be sets, then $\mathcal{F}(A_1, \ldots, A_m; B)$ denotes the set of all functions from $A_1 \times \cdots \times A_m$ to B.

2.9 Definition (Frame): A frame is a collection $\{\mathcal{D}_\tau\}_\tau$ of nonempty sets $\mathcal{D}_\tau$, one for each type symbol τ, such that $\mathcal{D}_o = \{\mathbf{T}, \mathbf{F}\}$ and $\mathcal{D}_{(\tau_1\times\cdots\times\tau_m\to\sigma)} \subseteq \mathcal{F}(\mathcal{D}_{\tau_1}, \ldots, \mathcal{D}_{\tau_m}; \mathcal{D}_\sigma)$. The members of $\mathcal{D}_o$ are called *truth values* and the members of $\mathcal{D}_\iota$ are called *individuals.*

2.10 Definition (Interpretation): An interpretation $\langle\{\mathcal{D}_\tau\}_\tau, \mathcal{J}\rangle$ of $\mathcal{L}^{\omega}$ consists of a frame and a function $\mathcal{J}$ that maps each constant of type τ of $\mathcal{L}^{\omega}$ to an element of $\mathcal{D}_\tau$.

2.11 Definition (Assignment): An assignment into a frame $\{D_\tau\}_\tau$ is a function ξ that maps each variable of type τ of $\mathcal{L}^{\omega}$ to an element of $\mathcal{D}_\tau$. An assignment into an interpretation is an assignment into the frame of the interpretation. In contexts where a particular interpretation is under discussion, it will be assumed that all assignments are into that interpretation unless otherwise indicated. Given an assignment ξ, a variable x_τ, and an element $d \in \mathcal{D}_\tau$, $\xi[x_\tau \leftarrow d]$ is defined as ξ except for x_τ where it is d.

2.12 Definition (Weak Interpretation): An interpretation $\mathcal{M} = \langle\{\mathcal{D}_\tau\}_\tau, \mathcal{J}\rangle$ is a weak interpretation (weak model, general model) for $\mathcal{L}^{\omega}$ ($\mathcal{L}^{\omega}_{\equiv}$) iff there is a binary function $\mathcal{V}^{\mathcal{M}}$ so that for every assignment ξ and term t of type τ, $\mathcal{V}^{\mathcal{M}}_\xi t \in \mathcal{D}_\tau$ and the following conditions hold:

1. for all variables x_τ, $\mathcal{V}^{\mathcal{M}}_\xi x_\tau = \xi x_\tau$
2. for all constants c_τ, $\mathcal{V}^{\mathcal{M}}_\xi c_\tau = \mathcal{J} c_\tau$
3. for composed terms $\mathcal{V}^{\mathcal{M}}_\xi(f_{(\tau_1\times\cdots\times\tau_m\to\sigma)}(t_{\tau_1}, \ldots, t_{\tau_m})) = \mathcal{V}^{\mathcal{M}}_\xi(f_{(\tau_1\times\cdots\times\tau_m\to\sigma)})(\mathcal{V}^{\mathcal{M}}_\xi t_{\tau_1}, \ldots, \mathcal{V}^{\mathcal{M}}_\xi t_{\tau_m})$
4. $\mathcal{V}^{\mathcal{M}}_\xi(\varphi \wedge \psi) = \mathcal{V}^{\mathcal{M}}_\xi\varphi \wedge \mathcal{V}^{\mathcal{M}}_\xi\psi$ *
5. $\mathcal{V}^{\mathcal{M}}_\xi(\neg\varphi) = \neg\mathcal{V}^{\mathcal{M}}_\xi\varphi$

*We use the connectives and quantifiers in a naive way at the meta-level.

6. $\mathcal{V}_\xi^{\mathcal{M}}(\forall x_\tau \varphi) = \forall d \in \mathcal{D}_\tau \mathcal{V}_{\xi[x_\tau \leftarrow d]}^{\mathcal{M}} \varphi$
7. for a model of $\mathcal{L}_{\equiv}^{\omega}$ we have additionally for all terms t_1, t_2 of type τ with $\tau \neq o$, $\mathcal{V}_\xi^{\mathcal{M}}(t_1 \equiv_{(\tau\times\tau\to o)} t_2) = \mathcal{V}_\xi^{\mathcal{M}}(t_1) \equiv_{\mathcal{D}_\tau} \mathcal{V}_\xi^{\mathcal{M}}(t_2)$*

2.13 Definition (Strong Interpretation): An interpretation $\mathcal{M} = \langle \{\mathcal{D}_\tau\}_\tau, \mathcal{J}\rangle$ is a strong interpretation (strong model, standard model) iff it is a weak interpretation and for all occuring types τ with $\tau = (\tau_1 \times \cdots \times \tau_m \to \sigma)$, $\mathcal{D}_\tau = \mathcal{F}(\mathcal{D}_{\tau_1}, \ldots, \mathcal{D}_{\tau_m}; \mathcal{D}_\sigma)$.

2.14 Remark:

- Every strong interpretation is in particular a weak interpretation.
- In order to fix a strong interpretation we only have to fix $\mathcal{D}_\iota$ and $\mathcal{J}$.

2.15 Remark: Of course equality at level n (for odd n) can be defined at level $n+1$ by G. W. LEIBNIZ' *identitas indiscernibilium*, that is, by the formula
$\forall x_\tau \forall y_\tau (x \equiv_{(\tau\times\tau\to o)} y) :\Longleftrightarrow (\forall P_{(\tau\to o)}\ P(x) \Longleftrightarrow P(y))$.
But if the underlying semantics is weak we would get by this definition a non-standard semantics for the equality predicate. For instance if we have two constants a_ι and b_ι and the equality predicate $\equiv_{(\iota\times\iota\to o)}$ defined by LEIBNIZ' identitas indiscernibilium in the signature of a logic. Now suppose we have the equality $a \equiv b$ as axiom, we could interprete the formula by $\mathcal{D}_\iota = \{1, 2\}$, $\mathcal{J}(a) = 1$, $\mathcal{J}(b) = 2$, and $\mathcal{D}_{(\iota\to o)}$ as the set that maps everything to T. Then we have $\mathcal{V}_\xi^{\mathcal{M}}(a \equiv b) = \mathtt{T}$ although a and b are interpreted by different elements. In other words we do not force that a and b are equal in all models by writing $a \equiv b$. In the logics $\mathcal{L}_{\equiv}$ we want the predicate constants $\equiv$ to be interpreted strongly. That is, if we have an equality like $a \equiv b$, then a and b must be mapped onto the same element in the corresponding universe.

2.16 Definition:

1. Let φ be a formula, $\mathcal{M}$ a weak (strong) interpretation. $\mathcal{M}$ is a weak (strong) model of φ if for every assignment ξ into $\mathcal{M}$, $\mathcal{V}_\xi^{\mathcal{M}}(\varphi) = \mathtt{T}$
2. A model for a set Γ of formulae is a model of each formula of Γ.
3. If every weak (strong) model of a formulae set Γ is also a weak (strong) model of a formula φ, we write $\Gamma \models_{weak} \varphi$ (respectively $\Gamma \models_{strong} \varphi$).

2.17 Example: Let P be a constant of type $(\iota \to o)$ and a be an object constant, that is, a constant of type ι. Then the formula $\varphi := \forall f_{(\iota\to\iota)} P(f(a)) \wedge \neg P(a)$ is unsatisfiable in the standard interpretation, because it is possible to choose the identity function for f. But we can find a weak model $\mathcal{M}$. For instance with $\mathcal{D}_\iota = \{1, 2\}$, $\mathcal{D}_{(\iota\to\iota)}$ consists of the one function that maps everything to 2, $\mathcal{J}(P)(1) = \mathtt{F}$, $\mathcal{J}(P)(2) = \mathtt{T}$, and $\mathcal{J}(a) = 1$, we get $\mathcal{V}_\xi^{\mathcal{M}}(\varphi) = \mathtt{T}$ for all assignments ξ.

2.18 Theorem: In $\mathcal{L}^1$ ($\mathcal{L}^1_{\equiv}$) for every weak model of a formula set there is a strong model with the same interpretation function $\mathcal{J}$.

Proof: Let Γ be a set of formulae in $\mathcal{L}^1$ and $\mathcal{M} = \langle \{\mathcal{D}_\tau\}_\tau, \mathcal{J}\rangle$ be a weak model of Γ. If we define $\overline{\mathcal{D}}_\iota$ as $\mathcal{D}_\iota$, $\overline{\mathcal{D}}_o$ as $\mathcal{D}_o$, and for all occuring types τ with $\tau = (\tau_1 \times \cdots \times \tau_m \to \sigma)$, $\overline{\mathcal{D}}_\tau := \mathcal{F}(\overline{\mathcal{D}}_{\tau_1}, \ldots, \overline{\mathcal{D}}_{\tau_m}; \overline{\mathcal{D}}_\sigma)$, then $\overline{\mathcal{M}} = \langle \{\overline{\mathcal{D}}_\tau\}_\tau, \mathcal{J}\rangle$ is a strong model of Γ. We have $\mathcal{D}_\tau \subseteq \overline{\mathcal{D}}_\tau$ for all types τ. With $\mathcal{V}^{\overline{\mathcal{M}}} = \mathcal{V}^{\mathcal{M}}$ the definition 2.12 is fulfilled automatically (because the interpretation function $\mathcal{J}$ is the same) except for 2.12.6. But 2.12.6 is satisfied, since in $\mathcal{L}^1$ we can quantify only over variables of type ι and $\overline{\mathcal{D}}_\iota = \mathcal{D}_\iota$. Therefore $\overline{\mathcal{M}}$ is a strong model of Γ. ■

*Here and in the following we use for all sets A, $\equiv_A$ as equality for elements in the set A.

3 Sorted Logics

In this section we introduce our target language, a many-sorted first order logic with equality predicates on all sorts. Let $SORT$ be a (finite) set of sorts. We define the signature $\mathcal{S}_{sort}$ of a logic in $\mathcal{L}^1_{sort}$ as a union of possibly empty sets $\mathcal{S}^{(s_1,\ldots,s_m):s}$ (m-ary function constants), $\mathcal{S}^{(s_1,\ldots,s_m)}$ (m-ary predicate constants), $\mathcal{S}^s_{const}$ (object constants), and the infinite countable sets $\mathcal{S}^s_{var}$ (object variables), where $s_1,\ldots,s_m,s \in SORT$. In each $\mathcal{S}^{(s,s)}$ we have the binary predicate symbol $\equiv^{(s,s)}$. We index the elements of $\mathcal{S}_{sort}$ sometimes by their sort. For instance a function symbol f of sort $(s_1,\ldots,s_m):s$ is written as $f^{(s_1,\ldots,s_m):s}$.

The order sorted logic of Oberschelp [11], operationalized by Walther [16] and Schmidt-Schauss [15] covers this simple situation and therefore the input language of a theorem prover like the Markgraf Karl Refutation Procedure [13] is well-suited for dealing with the now defined logic.

3.1 Definition (Sorted Terms):

- all elements of $\mathcal{S}^s_{const}$ and $\mathcal{S}^s_{var}$ are sorted terms of sort s
- if $t_1,\ldots,t_m$ are terms of sort $s_1,\ldots,s_m$, respectively, and if f is a function symbol of sort $(s_1,\ldots,s_m):s$, then $f(t_1,\ldots,t_m)$ is a term of sort s.

3.2 Definition (Sorted Formulae):

- if $t_1,\ldots,t_m$ are terms of sort $s_1,\ldots,s_m$ respectively, and if P is a predicate symbol of sort $(s_1,\ldots,s_m)$ then $P(t_1,\ldots,t_m)$ is an (atomic) formula.
- if t_1,t_2 are terms of sort s, then $(t_1 \equiv^{(s,s)} t_2)$ is an (atomic) formula.
- if φ and ψ are formulae and x^s is a variable of sort s, then $(\neg\varphi)$, $(\varphi\wedge\psi)$, and $(\forall x\varphi)$ are formulae.

3.3 Example: $\forall n^{nat} even(n) \wedge \forall h^{human} mortal(h)$ is a sorted formula.
$\forall n^{nat}\exists i^{integer} n \equiv i$ is not a well-defined sorted formula, because we have only flat sorts and so we cannot relate expressions of different sorts. Especially we cannot express the subset relation.

3.4 Definition (Semantics of Sorted Formulae): For every sort s in $\mathcal{S}$ we have a non-empty set $\mathcal{D}^s$. All $\mathcal{D}^s$ are disjoint. (We can choose them disjoint, because we have only unstructured sorts.) An interpretation of $\mathcal{L}_{sort}$ is a pair $\langle\{\mathcal{D}^s\}_s,\mathcal{J}\rangle$ where $\mathcal{J}$ maps each object-constant of sort s to an element of $\mathcal{D}^s$, each function constant of sort $(s_1,\ldots,s_m):s$ to an element of $\mathcal{F}(\mathcal{D}^{s_1},\ldots,\mathcal{D}^{s_m};\mathcal{D}^s)$, and each predicate constant of sort $(s_1,\ldots,s_m)$ to an element of $\mathcal{F}(\mathcal{D}^{s_1},\ldots,\mathcal{D}^{s_m};\mathcal{D}_o)$. $\mathcal{D}_o$ is defined as above as $\{\mathtt{T},\mathtt{F}\}$.
Let the assignments be defined as above. An interpretation $\mathcal{M}=\langle\{\mathcal{D}^s\}_s,\mathcal{J}\rangle$ is a (strong) interpretation for $\mathcal{L}^1_{sort}$ if there is a binary function $\mathcal{V}^{\mathcal{M}}$ so that for every assignment ξ and term t^s of sort s, $\mathcal{V}^{\mathcal{M}}_\xi(t^s)\in\mathcal{D}^s$, for every assignment ξ and for every formula φ, $\mathcal{V}^{\mathcal{M}}_\xi(\varphi)\in\mathcal{D}_o$ and the following conditions hold:

1. for all variables x^s, $\mathcal{V}^{\mathcal{M}}_\xi x^s = \xi x^s$
2. for all constants c^s, $\mathcal{V}^{\mathcal{M}}_\xi c^s = \mathcal{J}c^s$
3. for other terms $\mathcal{V}^{\mathcal{M}}_\xi(f^{(s_1,\ldots,s_m):s}(t^{s_1},\ldots,t^{s_m})) = \mathcal{V}^{\mathcal{M}}_\xi(f^{(s_1,\ldots,s_m):s})(\mathcal{V}^{\mathcal{M}}_\xi t^{s_1},\ldots,\mathcal{V}^{\mathcal{M}}_\xi t^{s_m})$
4. for non-equality atomic formulae
   $\mathcal{V}^{\mathcal{M}}_\xi(p^{(s_1,\ldots,s_m)}(t^{s_1},\ldots,t^{s_m})) = \mathcal{V}^{\mathcal{M}}_\xi(p^{(s_1,\ldots,s_m)})(\mathcal{V}^{\mathcal{M}}_\xi t^{s_1},\ldots,\mathcal{V}^{\mathcal{M}}_\xi t^{s_m})$
5. $\mathcal{V}^{\mathcal{M}}_\xi(t_1 \equiv^{(s,s)} t_2) = \mathcal{V}^{\mathcal{M}}_\xi(t_1) \equiv_{\mathcal{D}^s} \mathcal{V}^{\mathcal{M}}_\xi(t_2)$
6. $\mathcal{V}^{\mathcal{M}}_\xi(\varphi\wedge\psi) = \mathcal{V}^{\mathcal{M}}_\xi\varphi \wedge \mathcal{V}^{\mathcal{M}}_\xi\psi$
7. $\mathcal{V}^{\mathcal{M}}_\xi(\neg\varphi) = \neg\mathcal{V}^{\mathcal{M}}_\xi\varphi$
8. $\mathcal{V}^{\mathcal{M}}_\xi(\forall x^s\varphi) = \forall d\in\mathcal{D}^s \mathcal{V}^{\mathcal{M}}_{\xi[x\leftarrow d]}\varphi$

4 Logic Morphisms

Now we shall define those concepts that are necessary to describe the relation between formalizations in different logics. The important concepts are: logic, morphism, quasi-homomorphism, and soundness and completeness of a morphism.

4.1 Definition (Morphism of Logics): Let $\mathcal{F}^1$ and $\mathcal{F}^2$ be two logical systems ($\mathcal{L}^\omega$, $\mathcal{L}^\omega_\equiv$, $\mathcal{L}^n$, $\mathcal{L}^n_\equiv$, or $\mathcal{L}^1_{sort}$), then a morphism Θ is a mapping that maps the signature $\mathcal{S}$ of a logic $\mathcal{F}^1(\mathcal{S})$ in $\mathcal{F}^1$ to a signature of a logic $\mathcal{F}^2(\Theta(\mathcal{S}))$ in $\mathcal{F}^2$ and that maps every formula set in $\mathcal{F}^1(\mathcal{S})$ to a formula set in $\mathcal{F}^2(\Theta(\mathcal{S}))$.*

4.2 Definition (Soundness): Let Θ be a morphism from $\mathcal{F}^1$ to $\mathcal{F}^2$. Θ is called strongly (weakly) sound iff the following condition holds for every formula set Γ in $\mathcal{F}^1$:
if Γ has a strong (weak) model in $\mathcal{F}^1$ then there is a strong (weak) model of $\Theta(\Gamma)$ in $\mathcal{F}^2$.

4.3 Definition (Completeness): Let Θ be a morphism from $\mathcal{F}^1$ to $\mathcal{F}^2$. Θ is called strongly (weakly) complete iff the following condition holds for every formula set Γ in $\mathcal{F}^1$:
if $\Theta(\Gamma)$ has a strong (weak) model in $\mathcal{F}^2$ then there is a strong (weak) model of Γ in $\mathcal{F}^1$.

4.4 Definition (Quasi-Homomorphism): Let $\mathcal{F}^1(\mathcal{S}_1)$ and $\mathcal{F}^2(\mathcal{S}_2)$ be two logics. A mapping Θ that maps every fomula and every term of $\mathcal{F}^1(\mathcal{S}_1)$ to a formula respectively to a term of $\mathcal{F}^2(\mathcal{S}_2)$ is called a quasi-homomorphism iff the following conditions are satisfied:

1. For all terms:
   - 1.1 if x is a variable of $\mathcal{F}^1(\mathcal{S}_1)$ then $\Theta(x)$ is a variable of $\mathcal{F}^2(\mathcal{S}_2)$.
   - 1.2 if c is a constant of $\mathcal{F}^1(\mathcal{S}_1)$ then $\Theta(c)$ is a constant of $\mathcal{F}^2(\mathcal{S}_2)$.
   - 1.3 if $f(t_1,\ldots,t_m)$ is a term of $\mathcal{F}^1(\mathcal{S}_1)$ then $\Theta\left(f(t_1,\ldots,t_m)\right) = \vartheta\left(\Theta(f),\Theta(t_1),\ldots,\Theta(t_m)\right)$ with $\vartheta(a,a_1,\ldots,a_m) = \begin{cases} a(a_1,\ldots,a_m) & \text{or} \\ \alpha_a(a,a_1,\ldots,a_m) \end{cases}$
     The α have to be chosen appropriately out of $\mathcal{S}_2$, especially they have to be *new*, that is, there must be no element $\alpha' \in \mathcal{S}_1$ so that $\alpha_a = \Theta(\alpha')$. The case which is chosen can depend only on the a not on the $a_1,\ldots,a_m$. (α stands for *apply*.)
2. For all formulae φ_1,φ_2 and for all variables x:
   - 2.1 $\Theta(\varphi_1 \wedge \varphi_2) = \Theta(\varphi_1) \wedge \Theta(\varphi_2)$
   - 2.2 $\Theta(\neg\varphi) = \neg\Theta(\varphi)$
   - 2.3 $\Theta(\forall x\varphi) = \forall\Theta(x)\Theta(\varphi)$
3. All terms that are not formulae of $\mathcal{F}^1(\mathcal{S}_1)$ are mapped to terms that are not formulae of $\mathcal{F}^2(\mathcal{S}_2)$.

4.5 Remark:

- Of course we can extend the definition of a quasi-homomorphism to formula sets Γ by the requirement that the property must be fulfilled for every formula in Γ.
- We have excluded as quasi-homomorphism those mappings that map a formula like $P(a)$ onto a formula $P(a,a)$. That is, arguments cannot be doubled. We could allow this without losing anything essential in the sequel, but the proofs would become more teditious, without gaining really in expressive power.

*A formula is regarded as a formula set with one element. Especially we write $\Theta(\varphi)$ instead of $\Theta(\{\varphi\})$.

- If we translate into a first order logic, the first case for $\vartheta(a, a_1, \ldots, a_m)$ can only be chosen correctly if one is not going to quantify on a.

4.6 Lemma: If Θ is a strongly (weakly) sound quasi-homomorphism from $\mathcal{L}^n$ to $\mathcal{L}^1_{sort}$, $\vdash$ the derivability relation relative to a sound calculus of $\mathcal{L}^1_{sort}$, Γ a formula set and φ a formula in $\mathcal{L}^n$ with $\Theta(\Gamma) \vdash \Theta(\varphi)$, then $\Gamma \models_{strongly} \varphi$ (resp. $\Gamma \models_{weakly} \varphi$).

Proof: Because of $\Theta(\Gamma) \vdash \Theta(\varphi)$ we have that $\Theta(\Gamma) \cup \{\neg\Theta(\varphi)\}$ is unsatisfiable. Because of homomorphy in $\neg$ there is no model of $\Theta(\Gamma \cup \{\neg\varphi\})$. Hence by soundness there is no model of $\Gamma \cup \{\neg\varphi\}$. In other words every model of Γ is a model of φ. Because of theorem 2.18 this conclusion holds for both strong and weak models. ∎

4.7 Remark: We are especially interested in the situation where we can translate into $\mathcal{L}^1$ or $\mathcal{L}^1_{sort}$, because there are well-known complete calculi and strong theorem provers for these calculi. If we find a sound translation the lemma above guarantees that we can prove theorems in $\mathcal{L}^n$ by proving them in $\mathcal{L}^1$ or $\mathcal{L}^1_{sort}$. Strong completeness of such a translation is not obtainable because of GÖDEL'S incompleteness result, but a priori nothing speaks against weak completeness, that is, there might be a morphism Θ from $\mathcal{L}^n$ to $\mathcal{L}^1$ such that, if $\Gamma \models_{weak} \varphi$ then $\Theta(\Gamma) \vdash \Theta(\varphi)$ in $\mathcal{L}^1$.

$\mathcal{L}^1$ is not really appropriate as the target logic for a translation, but a sorted version $\mathcal{L}^1_{sort}$ is usually preferable. For a translation directly into $\mathcal{L}^1$ see BENTHEM and DOETS [3, p.316–320].

5 A SUFFICIENT CRITERION FOR SOUNDNESS

In this section we give a sufficient criterion for the soundness of translations of formulae of $\mathcal{L}^n$ onto formulae of $\mathcal{L}^1_{sort}$, which is strong enough to cover most requirements. In addition we give an example for such a sound translation.

5.1 Theorem: If Θ is an injective quasi-homomorphism from $\mathcal{L}^n(\mathcal{S})$ to $\mathcal{L}^1_{sort}(\mathcal{S}')$, then Θ is weakly sound.

Proof: Let $\mathcal{M}$ be a weak model of a formula set Γ, then $\mathcal{M} = \langle\{\mathcal{D}_\tau\}_\tau, \mathcal{J}\rangle$ is a weak model for any φ out of Γ, that is, $\mathcal{V}^{\mathcal{M}}_\xi \varphi = \mathtt{T}$ for every assignment ξ. We are going to construct a model $\hat{\mathcal{M}} = \langle\{\hat{\mathcal{D}}^{\tau'}\}_{\tau'}, \hat{\mathcal{J}}\rangle$ of $\Theta(\varphi)$. We define the sets $\hat{\mathcal{D}}^{\text{"}\tau\text{"}} := \mathcal{D}_\tau$. $\hat{\mathcal{J}}$ is defined as $\hat{\mathcal{J}}(\Theta(c)) := \mathcal{J}(c)$ for all constants c in $\mathcal{S}$. (Here and in the sequel we make use of the injectivity of Θ. In addition we use the fact that constants are mapped onto constants.) The assignments $\hat{\xi}$ are defined by $\hat{\xi}(\Theta(x)) := \xi(x)$. Because of $\hat{\mathcal{D}}^{\text{"}\tau\text{"}} = \mathcal{D}_\tau$ we get all assignments in this way. Recall that we have no function or predicate variables in $\mathcal{L}^1_{sort}$. We also use the fact that variables are mapped onto variables. For the functions $\alpha_a^{\text{"}\tau\text{"}}$ with $\tau = (\tau_1 \times \cdots \times \tau_m \to \sigma)$ we can define the interpretation so that it takes the interpretation of the first argument, which is a function, and applies it to the other arguments. We can do this, because these functions are new. Formally this interpretation can be written: for all $f \in \hat{\mathcal{D}}^{\text{"}\tau\text{"}}$, for all $x_1 \in \hat{\mathcal{D}}^{\text{"}\tau_1\text{"}}, \ldots, x_m \in \hat{\mathcal{D}}^{\text{"}\tau_m\text{"}}$ $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\alpha_f^{\text{"}\tau\text{"}})(f, t_1, \ldots, t_m) := f(t_1, \ldots, t_m)$. Note that $f \in \hat{\mathcal{D}}^{\text{"}\tau\text{"}} = \mathcal{D}_\tau$ is a function and hence applicable. By $f(t_1, \ldots, t_m)$ we mean the value and not the string as in the definition of its syntax.

Analogously we define the interpretation for the predicates $\alpha_p^{\text{"}\tau\text{"}}$ so that it takes the interpretation of the first argument, which is a predicate, and applies it to the other arguments:
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\alpha_p^{\text{"}\tau\text{"}})(p, x_1, \ldots, x_m) = p(x_1, \ldots, x_m)$.

Now $\hat{\mathcal{M}}$ is a model of $\Theta(\varphi)$, which is proved by induction on the construction of terms and formulae. Let $\mathcal{M}$ be a model of φ for all assignments ξ. We show that $\hat{\mathcal{M}}$ is a model of $\Theta(\varphi)$ for all assignment $\hat{\xi}$, that is, if $\mathcal{V}^{\mathcal{M}}_\xi(\varphi) = \mathtt{T}$ then $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\varphi)) = \mathtt{T}$. This can be proved by showing that for all terms and formulae $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}} \circ \Theta = \mathcal{V}^{\mathcal{M}}_\xi$.

For terms we have:

T1 For all variables x_τ, $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(x_\tau)) = \hat{\xi}(\Theta(x_\tau)) \stackrel{\text{def } \hat{\xi}}{=} \xi(x_\tau) = \mathcal{V}^{\mathcal{M}}_{\xi}(x_\tau)$.

T2 For all constants c_τ, $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(c_\tau)) = \hat{\mathcal{J}}(\Theta(c_\tau)) \stackrel{\text{def } \hat{\mathcal{J}}}{=} \mathcal{J}(c_\tau) = \mathcal{V}^{\mathcal{M}}_{\xi}(c_\tau)$.

T3 For all composed terms that start with an m-ary function term f so that $\vartheta(f, t_1, \ldots, t_m)$ is defined as $f(t_1, \ldots, t_m)$ we have:
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(f_\tau(t_1, \ldots, t_m))) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\vartheta(\Theta(f), \Theta(t_1), \ldots, \Theta(t_m))) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(f)(\Theta(t_1), \ldots, \Theta(t_m))) \stackrel{\text{def}}{=}$
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}\Theta(f)(\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}\Theta(t_1), \ldots, \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}\Theta(t_m)) \stackrel{\text{Ind.hyp}}{=} \mathcal{V}^{\mathcal{M}}_{\xi}(f)(\mathcal{V}^{\mathcal{M}}_{\xi}(t_1), \ldots, \mathcal{V}^{\mathcal{M}}_{\xi}(t_m)) =$
$\mathcal{V}^{\mathcal{M}}_{\xi}(f_\tau(t_1, \ldots, t_m))$.

T4 For all composed terms that start with an m-ary function term f so that $\vartheta(f, t_1, \ldots, t_m)$ is defined as $\alpha_f{}^{"\tau"}(f, t_1, \ldots, t_m)$ we have:
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(f_\tau(t_1, \ldots, t_m))) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\vartheta(\Theta(f), \Theta(t_1), \ldots, \Theta(t_m))) =$
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\alpha_f{}^{"\tau"}(\Theta(f), \Theta(t_1), \ldots, \Theta(t_m))) \stackrel{\text{def}}{=} \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\alpha_f{}^{"\tau"})(\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}\Theta(f), \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}\Theta(t_1), \ldots, \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}\Theta(t_m)) \stackrel{\text{Ind.hyp}}{=}$
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\alpha_f{}^{"\tau"})(\mathcal{V}^{\mathcal{M}}_{\xi}(f), \mathcal{V}^{\mathcal{M}}_{\xi}(t_1), \ldots, \mathcal{V}^{\mathcal{M}}_{\xi}(t_m)) \stackrel{\text{def}}{=} \mathcal{V}^{\mathcal{M}}_{\xi}(f)(\mathcal{V}^{\mathcal{M}}_{\xi}(t_1), \ldots, \mathcal{V}^{\mathcal{M}}_{\xi}(t_m)) =$
$\mathcal{V}^{\mathcal{M}}_{\xi}(f_\tau(t_1, \ldots, t_m))$.

For formulae we get:

F1 An atomic formula is a special term, hence we have already proved the required property above.

F2 For a conjunction we have:
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\varphi_1 \wedge \varphi_2)) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\varphi_1) \wedge \Theta(\varphi_2)) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\varphi_1)) \wedge \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\varphi_2)) \stackrel{\text{Ind.hyp}}{=}$
$\mathcal{V}^{\mathcal{M}}_{\xi}(\varphi_1) \wedge \mathcal{V}^{\mathcal{M}}_{\xi}(\varphi_2) = \mathcal{V}^{\mathcal{M}}_{\xi}(\varphi_1 \wedge \varphi_2)$.

F3 For a negation we have:
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\neg\varphi)) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\neg\Theta(\varphi)) = \neg\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\varphi)) \stackrel{\text{Ind.hyp}}{=} \neg\mathcal{V}^{\mathcal{M}}_{\xi}(\varphi) = \mathcal{V}^{\mathcal{M}}_{\xi}(\neg\varphi)$.

F4 For a quantification we have:
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\forall x_\tau \varphi)) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\forall x^{"\tau"}\Theta(\varphi)) = \forall d \in \hat{\mathcal{D}}^{"\tau"}\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}[x^{"\tau"} \leftarrow d]}(\Theta(\varphi)) =$
$\forall d \in \mathcal{D}_\tau \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}[\Theta(x_\tau) \leftarrow d]}(\Theta(\varphi)) \stackrel{\text{Ind.hyp}}{=} \forall d \in \mathcal{D}_\tau \mathcal{V}^{\mathcal{M}}_{\xi[x_\tau \leftarrow d]}(\varphi) = \mathcal{V}^{\mathcal{M}}_{\xi}(\forall x \varphi)$.

Here we use that $\hat{\xi} \circ \Theta = \xi$ for all assignments and hence $\hat{\xi}[\Theta(x_\tau) \leftarrow d] \circ \Theta = \xi[x_\tau \leftarrow d]$.

Hence for all formulae we have shown that $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}} \circ \Theta = \mathcal{V}^{\mathcal{M}}_{\xi}$. So we can conclude: if $\mathcal{V}^{\mathcal{M}}_{\xi}(\varphi) = \mathtt{T}$ then $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\varphi)) = \mathtt{T}$. ∎

5.2 Theorem: If Θ is an injective quasi-homomorphism from $\mathcal{L}^n(\mathcal{S})$ to $\mathcal{L}^1_{sort}(\mathcal{S}')$, then Θ is strongly sound.

Proof: If there is a strong model of a formula set Γ in $\mathcal{L}^n(\mathcal{S})$ then this model is also a weak model. By the previous theorem there is hence a weak model of $\Theta(\Gamma)$ in $\mathcal{L}^1_{sort}(\mathcal{S}')$. By a sorted version of theorem 2.18 there is also a strong model of $\Theta(\Gamma)$. ∎

5.3 Example: Let us see how to translate the predicative definition of a group into first order logic. We drop the type information for readability, *group* is of type $(\iota \times (\iota \times \iota \times \iota \to o) \to o)$, G of type ι, $+$ of type $(\iota \times \iota \times \iota \to o)$, $-$ of type $(\iota \times \iota \to o)$, and so on. In the translation this transforms into the sorts $("\iota", "(\iota \times \iota \times \iota \to o)")$ for *group*, and so on. A group can be defined as follows:

1. $\forall G, + \; group(G, +) \iff associative(G, +) \wedge$
$\exists 0 \; (0 \in G \wedge neutral\_element(G, +, 0) \wedge$
$\exists - \; inverse(G, +, 0, -))$

2. $\forall G, + \; associative(G, +) \iff$
$\forall u, v, w, x, y, z \; u, v, w, x, y, z \in G \wedge +(x, y, u) \wedge +(u, z, w) \wedge +(y, z, v) \implies$
$+(x, v, w)$

3. $\forall G,+,0\ \ neutral\_element(G,+,0) \iff \forall x\ \ x \in G \Longrightarrow +(x,0,x)\ \wedge\ +(0,x,x)$

4. $\forall G,+,0,-\ \ inverse(G,+,0,-) \iff \forall x,y\ \ x,y \in G\ \wedge\ -(x,y) \Longrightarrow +(x,y,0)\ \wedge\ +(y,x,0)$

This formula set is a subset of $\mathcal{L}^3$. Now we give a translation into a formula set of $\mathcal{L}^1_{sort}$. The signatures are obvious, hence ommitted. The translation is sound, because it is an injective quasi-homomorphism.

1. $\forall G,+\ \ group(G,+) \iff associative(G,+)\ \wedge$
$\exists 0\ \ (0 \in G\ \wedge\ neutral\_element(G,+,0)\ \wedge$
$\exists -\ \ inverse(G,+,0,-))$

2. $\forall G,+\ \ associative(G,+) \iff$
$\forall u,v,w,x,y,z\ \ u \in G\ \wedge\ v \in G\ \wedge\ w \in G\ \wedge\ x \in G\ \wedge\ y \in G\ \wedge\ z \in G\ \wedge$
$\boldsymbol{\alpha}^{"(\iota\times\iota\times\iota\to o)"}(+,x,y,u)\ \wedge\ \boldsymbol{\alpha}^{"(\iota\times\iota\times\iota\to o)"}(+,u,z,w)\ \wedge$
$\boldsymbol{\alpha}^{"(\iota\times\iota\times\iota\to o)"}(+,y,z,v)\ \Longrightarrow$
$\boldsymbol{\alpha}^{"(\iota\times\iota\times\iota\to o)"}(+,x,v,w)$

3. $\forall G,+,0\ \ neutral\_element(G,+,0) \iff \forall x\ \ x \in G \Longrightarrow$
$\boldsymbol{\alpha}^{"(\iota\times\iota\times\iota\to o)"}(+,x,0,x)\ \wedge\ \boldsymbol{\alpha}^{"(\iota\times\iota\times\iota\to o)"}(+,0,x,x)$

4. $\forall G,+,0,-\ \ inverse(G,+,0,-) \iff \forall x,y\ \ x \in G\ \wedge\ y \in G\ \wedge\ \boldsymbol{\alpha}^{"(\iota\times\iota\to o)"}(-,x,y) \Longrightarrow$
$\boldsymbol{\alpha}^{"(\iota\times\iota\times\iota\to o)"}(+,x,y,0)\ \wedge$
$\boldsymbol{\alpha}^{"(\iota\times\iota\times\iota\to o)"}(+,y,x,0)$

This translation is clumsy, because we cannot use equality; a translation with equality can be found in example 7.3.

5.4 Remark: Note that the formulae that are obtained by these translations are not essentially more difficult as the original, the structure of the formulae (number and position of quantifiers and junctors) is respected. In the image the terms are never more nested than in the original. The only thing that can change, is that the number of arguments in a term is increased by one.

5.5 Remark: For the final proof presentation proofs can easily be translated back, because the mappings Θ are injective. In other words if we have a first order calculus then this calculus provides a calculus for $\mathcal{L}^n$ by Θ^{-1}.

6 The Standard Translation From Unsorted Higher Order Logic to Sorted First Order Logic

Now we want to define morphisms $\hat{\Theta}_n$ from $\mathcal{L}^n$ to $\mathcal{L}^1_{sort}$ which are not only sound but also complete. We define the morphisms for odd n, for even n they are obtained as the restriction of the next higher odd n, that is $\hat{\Theta}_{2n} := \hat{\Theta}_{2n+1}\ |_{\mathcal{L}^{2n}}$. The morphisms $\hat{\Theta}$ are defined as $\hat{\Theta}(\varphi) = \hat{\Theta}'(\varphi) \cup \%AXIOMS$, where $\hat{\Theta}'(\varphi)$ is a quasi-homomorphism and $\%AXIOMS$ is the set of extensionality axioms which depends only on the signature of the logic. In the following we drop the index n. Again we abbreviate *apply* as $\boldsymbol{\alpha}$.

6.1 Definition (Standard Translation $\hat{\Theta}_{2n-1}$): Let $\mathcal{S}^{2n-1} = \bigcup_\tau \mathcal{S}_\tau$ be the signature of a logic in $\mathcal{L}^{2n-1}$. We define a signature $\mathcal{S}_{sort}$ of a logic in $\mathcal{L}^1_{sort}$ by assigning to each predicate constant of order n, arity m, and type $\tau = (\tau_1 \times \cdots \times \tau_m \to o)$ a predicate constant of order 1, arity m (that is, of type $(\iota \times \cdots \times \iota \to o)$)* and sort $("\tau_1", \ldots, "\tau_m")$**. All constants and variables of order less than n and of a type σ are mapped onto constants and variables of type ι and sort "σ". Because we assumed all members in $\mathcal{S}^{2n-1}$ to be disjoint, we can use the same names for the images.

*Recall: In $\mathcal{L}^{2n-1}$ there are no function constants of order n.

**By "τ" we mean the string after expanding the abbreviation for τ, for instance, if $\tau = (\iota \times \iota \to o)$ then "τ" is "$(\iota \times \iota \to o)$".

In addition we have in $\mathcal{S}_{sort}$ for each type τ of order less than n with $\tau = (\tau_1 \times \cdots \times \tau_m \to o)$ a new $(m+1)$-ary predicate constant $\boldsymbol{\alpha}^{"\tau"}$ of sort $("\tau", "\tau_1", \ldots, "\tau_m")$ and for each type τ of order less than n with $\tau = (\tau_1 \times \cdots \times \tau_m \to \sigma)$, $\sigma \neq o$ a new $(m+1)$-ary function constant $\boldsymbol{\alpha}^{"\tau"}$ of sort $("\tau", "\tau_1", \ldots, "\tau_m") : "\sigma"$.

Now we are going to define a quasi-homomorphism $\hat{\Theta}'$. For terms it is defined inductively by:

T1 for all variables x_τ, $\hat{\Theta}'(x_\tau) = x^{"\tau"}$

T2 for all constants c_τ of order equal n with $\tau = (\tau_1 \times \cdots \times \tau_m \to o)$, $\hat{\Theta}'(c_\tau) = c^{("\tau_1", \ldots, "\tau_m")}$

T3 for all constants c_τ of order less than n, $\hat{\Theta}'(c_\tau) = c^{"\tau"}$

T4 For a term with an m-ary function term f of type τ as top expression we define $\hat{\Theta}'(f(t_1, \ldots, t_m)) = \boldsymbol{\alpha}^{"\tau"}(\hat{\Theta}'(f), \hat{\Theta}'(t_1), \ldots, \hat{\Theta}'(t_m))$

For formulae we define $\hat{\Theta}'$ inductively by:

F1 For an atomic formula with predicate constant p of order n as top expression we define $\hat{\Theta}'(p(t_1, \ldots, t_m)) = \hat{\Theta}'(p)(\hat{\Theta}'(t_1), \ldots, \hat{\Theta}'(t_m))$

F2 For a term with an m-ary predicate term p of type τ and order less than n as top expression we define $\hat{\Theta}'(p(t_1, \ldots, t_m)) = \boldsymbol{\alpha}^{"\tau"}(\hat{\Theta}'(p), \hat{\Theta}'(t_1), \ldots, \hat{\Theta}'(t_m))$

F3 For a conjunction we define $\hat{\Theta}'(\varphi_1 \wedge \varphi_2) = \hat{\Theta}'(\varphi_1) \wedge \hat{\Theta}'(\varphi_2)$

F4 For a negation we define $\hat{\Theta}'(\neg\varphi) = \neg\hat{\Theta}'(\varphi)$

F5 For a quantified formula we define $\hat{\Theta}'(\forall x\varphi) = \forall\hat{\Theta}'(x)\hat{\Theta}'(\varphi)$

$\%AXIOMS$ is the set consisting of the following formulae of $\mathcal{L}^1_{sort}$:

A1 For every function constant $\boldsymbol{\alpha}^{"\tau"}$ with $\tau = (\tau_1 \times \cdots \times \tau_m \to \sigma)$, $\sigma \neq o$:
$\forall f^{"\tau"}\forall g^{"\tau"}(\forall x_1^{"\tau_1"}, \ldots, \forall x_m^{"\tau_m"}\ \boldsymbol{\alpha}^{"\tau"}(f, x_1, \ldots, x_m) \equiv^{("\sigma","\sigma")} \boldsymbol{\alpha}^{"\tau"}(g, x_1, \ldots, x_m)) \Longrightarrow f \equiv^{("\tau","\tau")} g$

A2 For every predicate constant $\boldsymbol{\alpha}^{"\tau"}$ with $\tau = (\tau_1 \times \cdots \times \tau_m \to o)$:
$\forall p^{"\tau"}\forall q^{"\tau"}(\forall x_1^{"\tau_1"}, \ldots, \forall x_m^{"\tau_m"}\boldsymbol{\alpha}^{"\tau"}(p, x_1, \ldots, x_m) \iff \boldsymbol{\alpha}^{"\tau"}(q, x_1, \ldots, x_m)) \Longrightarrow p \equiv^{("\tau","\tau")} q$

We define $\hat{\Theta}(\varphi) = \hat{\Theta}'(\varphi) \cup \%AXIOMS$. Analogously for formula sets $\hat{\Theta}(\Gamma) = \hat{\Theta}'(\Gamma) \cup \%AXIOMS$.

6.2 Remark: It should become obvious now, why we excluded types like $(o \to o)$: Let P be a predicate of this type, Q be a predicate of type $(\iota \to o)$, and c be a constant of type ι. Then $\hat{\Theta}'(P(Q(c)) \wedge Q(c))$ would be $\boldsymbol{\alpha}^{"(o\to o)"}(P, \boldsymbol{\alpha}^{"(\iota\to o)"}(Q, c)) \wedge \boldsymbol{\alpha}^{"(\iota\to o)"}(Q, c)$ or $P(\boldsymbol{\alpha}^{"(\iota\to o)"}(Q, c)) \wedge \boldsymbol{\alpha}^{"(\iota\to o)"}(Q, c)$ which is not well-formed, because $\boldsymbol{\alpha}^{"(\iota\to o)"}(Q, c)$ has to be a formula and a term at once. Even worse in general a *uniform* (quasi-homomorphic) translation is not possible, because $Q(c)$ must be translated in the first case to a term and in the second to a formula, what is not allowed in first order logic. I think that this example is also a counterexample for the correctness of the translation given by BENTHEM and DOETS [3] for a language without function symbols.

A possible translation of the unrestricted typed higher order logic has also to provide a translation of formulae of the kind $P(Q(c)) \wedge Q(c)$. This is possible by having only function symbols $\boldsymbol{\alpha}^{"\tau"}$ and translating all other symbols into object variables or object constants. Especially the junctor "$\wedge$" has also to be translated to a constant. A possible translation would be:
$\boldsymbol{\alpha}^{"(o\to o)"}(\wedge, \boldsymbol{\alpha}^{"(o\to o)"}(P, \boldsymbol{\alpha}^{"(\iota\to o)"}(Q, c)), \boldsymbol{\alpha}^{"(\iota\to o)"}(Q, c)) \equiv \texttt{TRUE}$. That is, the whole problem is encoded into an equality problem. In order to gain completeness it would be necessary to add axioms for the junctor "$\wedge$".

We have not defined translations of arbitrary formula sets of $\mathcal{L}^\omega$. For instance with the unary predicate symbols $P^1_{(\iota\to o)}$, $P^2_{((\iota\to o)\to o)}$, $P^3_{(((\iota\to o)\to o)\to o)}, \ldots$, the formula set $\Gamma = \bigcup_{n\geq 1}\{P^{n+1}(P^n)\}$ is not

translatable. Of course our mappings $\hat{\Theta}_n$ could be extended to a mapping $\hat{\Theta}_\omega$ in such a way that we have as predicates only the $\boldsymbol{\alpha}^{"\tau"}$. We have not done this, because in all practical cases only finitely many formulae are involved and so we can have a translation $\hat{\Theta}_n$. This gives a translation that preserves the property of being a predicate for as many symbols as possible.

6.3 Lemma: $\hat{\Theta}'_{2n-1}$ is a quasi-homomorphism from $\mathcal{L}^{2n-1}(\mathcal{S})$ to $\mathcal{L}^1_{sort}(\hat{\Theta}(\mathcal{S}))$.

Proof: We have to prove that for every formula φ in a logic $\mathcal{L}^{2n-1}(\mathcal{S})$, $\hat{\Theta}'_{2n-1}(\varphi)$ is a well-sorted formula of $\mathcal{L}^1_{sort}(\hat{\Theta}_{2n-1}(\mathcal{S}))$. Instead of $\hat{\Theta}'_{2n-1}$ we write shortly $\hat{\Theta}'$. We prove this by induction on the construction of terms and formulae:

- If x is a variable of type τ, then $\hat{\Theta}'(x)$ is a variable of sort $"\tau"$ hence well-sorted.
- If c is a constant of type τ and order less then n, then $\hat{\Theta}'(c)$ is a constant of sort $"\tau"$ hence well-sorted.
- If c is a constant of type $\tau = (\tau_1 \times \cdots \times \tau_m \to o)$ and order equal n, then $\hat{\Theta}'(c)$ is a constant of sort $("\tau_1", \ldots, "\tau_m")$ hence well-sorted.
- If $f_\tau, t_{\tau_1}, \ldots, t_{\tau_m}$ are terms of the types indicated by their subscripts with $\tau = (\tau_1 \times \cdots \times \tau_m \to \sigma)$ and order less then n, then $\hat{\Theta}'(f_\tau(t_{\tau_1}, \ldots, t_{\tau_m})) = \boldsymbol{\alpha}^{"\tau"}(\hat{\Theta}'(f), \hat{\Theta}'(t_{\tau_1}), \ldots, \hat{\Theta}'(t_{\tau_m}))$ where $\boldsymbol{\alpha}^{"\tau"}$ has sort $("(\tau_1 \times \cdots \times \tau_m \to \sigma)", "\tau_1", \ldots, "\tau_m") : "\sigma"$, $\hat{\Theta}'(f)$ has sort $"\tau"$, and the $\hat{\Theta}'(t_{\tau_i})$ have sorts $"\tau_i"$, hence the term is well-sorted.
- If $p_\tau, t_{\tau_1}, \ldots, t_{\tau_m}$ are terms of the types indicated by their subscripts with $\tau = (\tau_1 \times \cdots \times \tau_m \to o)$ and order n, then $\hat{\Theta}'(p_\tau(t_{\tau_1}, \ldots, t_{\tau_m})) = (\hat{\Theta}'(p)(\hat{\Theta}'(t_{\tau_1}), \ldots, \hat{\Theta}'(t_{\tau_m}))$, where $\hat{\Theta}'(p)$ has sort $("\tau_1", \ldots, "\tau_m")$, and the $\hat{\Theta}'(t_{\tau_i})$ have sorts $"\tau_i"$, hence the whole formula is well-sorted.
- If $p_\tau, t_{\tau_1}, \ldots, t_{\tau_m}$ are terms of the types indicated by their subscripts with $\tau = (\tau_1 \times \cdots \times \tau_m \to o)$ and order less then n, then $\hat{\Theta}'(p_\tau(t_{\tau_1}, \ldots, t_{\tau_m})) = \boldsymbol{\alpha}^{"\tau"}(\hat{\Theta}'(p), \hat{\Theta}'(t_{\tau_1}), \ldots, \hat{\Theta}'(t_{\tau_m}))$, where $\boldsymbol{\alpha}^{"\tau"}$ has sort $("(\tau_1 \times \cdots \times \tau_m \to o)", "\tau_1", \ldots, "\tau_m")$, $\hat{\Theta}'(p)$ has sort $"\tau"$, and the $\hat{\Theta}'(t_{\tau_i})$ have sorts $"\tau_i"$, hence the whole formula is well-sorted.

This shows the well-sortedness for atomic formulae, for composed formulae this property is immediate. The properties of a quasi-homomorphism follow trivially from the definition of $\hat{\Theta}'$. Constants and variables are mapped onto constants respectively variables. Terms fulfill the required property and the homomorphy of the junctors and the quantifier is also given. ∎

6.4 Remark: It is easy to show inductively that $\hat{\Theta}'_{2n-1}$ is an injective mapping on $\mathcal{L}^{2n-1}(\mathcal{S})$. Hence it is a bijective mapping from $\mathcal{L}^{2n-1}(\mathcal{S})$ to $\hat{\Theta}'_{2n-1}(\mathcal{L}^{2n-1}(\mathcal{S}))$.

6.5 Theorem: $\hat{\Theta}$ is weakly sound.

Proof: Let $\mathcal{M} = \langle \{\mathcal{D}_\tau\}_\tau, \mathcal{J} \rangle$ be a weak model of a formula set Γ, hence $\mathcal{M}$ is a weak model for any φ out of Γ, that is, $\mathcal{V}^{\mathcal{M}}_\xi \varphi = \mathtt{T}$ for every assignment ξ. We are going to show that $\hat{\mathcal{M}} = \langle \{\hat{\mathcal{D}}^{\tau'}\}_{\tau'}, \hat{\mathcal{J}} \rangle$ (as in theorem 5.1) is a model of $\hat{\Theta}(\varphi)$. By theorem 5.1 and remark 6.4 we have that $\hat{\Theta}'$ is sound. So it remains to be shown that for all elements φ in $\%AXIOMS$ we have $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\varphi) = \mathtt{T}$. That is, we have to show that

$$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}} \begin{pmatrix} \forall f^{"\tau"} \forall g^{"\tau"} (\forall x_1^{"\tau_1"}, \ldots, \forall x_m^{"\tau_m"} \\ \boldsymbol{\alpha}^{"\tau"}(f, x_1, \ldots, x_m) \equiv^{("\sigma","\sigma")} \boldsymbol{\alpha}^{"\tau"}(g, x_1, \ldots, x_m)) \Longrightarrow f \equiv^{("\tau","\tau")} g \end{pmatrix} = \mathtt{T}.$$

Therefore it is necessary to prove that for all F, G in $\hat{\mathcal{D}}^{"\tau"}$, we get $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}[f,g \leftarrow F,G]}(f) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}[f,g \leftarrow F,G]}(g)$ (that is, $F = G$), if for all $X_1 \in \hat{\mathcal{D}}^{"\tau_1"}, \ldots, X_m \in \hat{\mathcal{D}}^{"\tau_m"}$

$$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}[f,g,x_1,\ldots,x_m \leftarrow F,G,X_1,\ldots,X_m]}(\boldsymbol{\alpha}^{"\tau"}(f, x_1, \ldots, x_m)) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}[f,g,x_1,\ldots,x_m \leftarrow F,G,X_1,\ldots,X_m]}(\boldsymbol{\alpha}^{"\tau"}(g, x_1, \ldots, x_m)).$$

By the definition of $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}[f,g,x_1,\ldots,x_m \leftarrow F,G,X_1,\ldots,X_m]}$ this is equivalent to $F(X_1, \ldots, X_m) = G(X_1, \ldots, X_m)$. Because two functions are the same, if they have the same values on all arguments we get $F = G$, what was to prove.

The axioms for the predicates can be proved to be true analogously. ∎

6.6 Remark: $\hat{\Theta}$ is strongly sound, analogously to theorem 5.2.

6.7 Theorem: $\hat{\Theta}$ is weakly complete.

Proof: Let Γ be a formula set in $\mathcal{L}^{2n-1}(\mathcal{S})$. Let $\mathcal{M}$ be a weak model of $\hat{\Theta}(\Gamma)$. Then $\mathcal{M}$ is a model of $\hat{\Theta}(\varphi)$ for every formula φ in Γ. Let $\mathcal{M}$ be $\langle\{\mathcal{D}^s\}_s, \mathcal{J}\rangle$ and ξ be an arbitrary assignment. Then we have $\mathcal{V}_\xi^{\mathcal{M}}(\hat{\Theta}(\varphi)) = \mathtt{T}$. We want to construct a model $\check{\mathcal{M}}$ of φ, so that for all assignments $\check{\xi}$ we have $\mathcal{V}_{\check{\xi}}^{\check{\mathcal{M}}}(\varphi) = \mathtt{T}$. Therefore we define $\check{\mathcal{D}}_\iota := \mathcal{D}^{"\iota"}$ and $\check{\mathcal{D}}_o := \{\mathtt{T}, \mathtt{F}\}$. For all other types τ with $\tau = (\tau_1 \times \cdots \times \tau_m \to \sigma)$ we have to define $\check{\mathcal{D}}_\tau \subseteq \mathcal{F}(\check{\mathcal{D}}_{\tau_1}, \ldots, \check{\mathcal{D}}_{\tau_m}; \check{\mathcal{D}}_\sigma)$. We do it by inductively defining injective functions $\natural_\tau$ from $\mathcal{D}^{"(\tau_1 \times \cdots \times \tau_m \to \sigma)"}$ to $\mathcal{F}(\check{\mathcal{D}}_{\tau_1}, \ldots, \check{\mathcal{D}}_{\tau_m}; \check{\mathcal{D}}_\sigma)$ and setting $\check{\mathcal{D}}_\tau := \natural_\tau(\mathcal{D}^{"\tau"})$. Hence $\natural_\tau$ is a bijective function from $\mathcal{D}^{"\tau"}$ to $\check{\mathcal{D}}_\tau$.*

We define $\natural_\tau$ as bijective functions inductively:

1. $\natural_\iota : \mathcal{D}^{"\iota"} \to \check{\mathcal{D}}_\iota$ as the identity mapping (This function is obviously bijective).
2. Let $\natural_{\tau_i}$ and $\natural_\sigma$ be defined for $\mathcal{D}^{"\tau_1"}, \ldots, \mathcal{D}^{"\tau_m"}$, and $\mathcal{D}^{"o"}$. We are going to define a function $\natural_\tau$ with $\tau = (\tau_1 \times \cdots \times \tau_m \to \sigma)$, $\sigma \neq o$, for $\mathcal{D}^{"\tau"}$. For all $x \in \mathcal{D}^{"\tau"}$ $\natural_\tau(x)$ is defined as the function $\natural_\tau(x)(\check{x}_1, \ldots, \check{x}_m) := \natural_\sigma(\mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(x, \natural_{\tau_1}^{-1}(\check{x}_1), \ldots, \natural_{\tau_m}^{-1}(\check{x}_m)))$ for all $\check{x}_1 \in \check{\mathcal{D}}_{\tau_1}, \ldots, \check{x}_m \in \check{\mathcal{D}}_{\tau_m}$

   The following diagram may help to see the involved mappings at a glance:

$$\begin{array}{ccccccccc} \mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"}) : & \mathcal{D}^{"\tau"} & \times & \mathcal{D}^{"\tau_1"} & \times \cdots \times & \mathcal{D}^{"\tau_m"} & \longrightarrow & \mathcal{D}^{"\sigma"} \\ & \Big\downarrow{\scriptstyle !}\Big\uparrow{\scriptstyle \natural_\tau} & & \Big\uparrow{\scriptstyle \natural_{\tau_1}^{-1}} & & \Big\uparrow{\scriptstyle \natural_{\tau_m}^{-1}} & & \Big\downarrow{\scriptstyle \natural_\sigma} \\ & \check{\mathcal{D}}_\tau & \hookrightarrow & \mathcal{F}(\check{\mathcal{D}}_{\tau_1} & , \ldots , & \check{\mathcal{D}}_{\tau_m} & ; & \check{\mathcal{D}}_\sigma) \end{array}$$

   In order to show the injectivity of $\natural_\tau$ we use that we have in $\%AXIOMS$ the formula
   $\forall f^{"\tau"} \forall g^{"\tau"} (\forall x_1^{"\tau_1"}, \ldots, \forall x_m^{"\tau_m"} \boldsymbol{\alpha}^{"\tau"}(f, x_1, \ldots, x_m) \equiv^{("\sigma","\sigma")} \boldsymbol{\alpha}^{"\tau"}(g, x_1, \ldots, x_m)) \Longrightarrow f \equiv^{("\tau","\tau")} g$

   Therefore we have in a model for all x, x' in $\mathcal{D}^{"\tau"}$

$$\forall y_1 \in \mathcal{D}^{"\tau_1"}, \ldots, \forall y_m \in \mathcal{D}^{"\tau_m"} \mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(x, y_1, \ldots, y_m) \equiv_{\mathcal{D}^{"\sigma"}} \mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(x', y_1, \ldots, y_m) \Longrightarrow x \equiv_{\mathcal{D}^{"\tau"}} x' \qquad (*)$$

   Let $\natural_\tau(x) \equiv_{\check{\mathcal{D}}_\tau} \natural_\tau(x')$ for arbitrary x and x' in $\mathcal{D}^{"\tau"}$. Then we have by definition for all $\check{x}_1 \in \check{\mathcal{D}}_{\tau_1}, \ldots, \check{x}_m \in \check{\mathcal{D}}_{\tau_m}$
   $\natural_\sigma \mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(x, \natural_{\tau_1}^{-1}(\check{x}_1), \ldots, \natural_{\tau_m}^{-1}(\check{x}_m)) \equiv_{\check{\mathcal{D}}_\sigma} \natural_\sigma \mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(x', \natural_{\tau_1}^{-1}(\check{x}_1), \ldots, \natural_{\tau_m}^{-1}(\check{x}_m))$. Since the mappings $\natural_{\tau_1}, \ldots, \natural_{\tau_m}, \natural_\sigma$ are all bijective, we get for all $y_1 \in \mathcal{D}^{"\tau_1"}, \ldots, y_m \in \mathcal{D}^{"\tau_m"}$:
   $\mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(x, y_1, \ldots, y_m) \equiv_{\mathcal{D}^{"\sigma"}} \mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(x', y_1, \ldots, y_m)$. Because by $(*)$ $x \equiv_{\mathcal{D}^{"\tau"}} x'$, the injectivity is shown. Since the surjectivity is given by definition, we have proved that $\natural_\tau$ is bijective.
3. Let $\natural_{\tau_i}$ be defined for $\mathcal{D}^{"\tau_1"}, \ldots, \mathcal{D}^{"\tau_m"}$. We define a function $\natural_\tau$ (for order of τ is equal to n) with $\tau = (\tau_1 \times \cdots \times \tau_m \to o)$ for $\mathcal{D}^{("\tau_1",\ldots,"\tau_m")}$. For all $p \in \mathcal{D}^{("\tau_1",\ldots,"\tau_m")}$ $\natural_\tau(p)$ is defined as the predicate $\natural_\tau(p)(\check{x}_1, \ldots, \check{x}_m) := p(\natural_{\tau_1}^{-1}(\check{x}_1), \ldots, \natural_{\tau_m}^{-1}(\check{x}_m))$ for all $\check{x}_1 \in \check{\mathcal{D}}_{\tau_1}, \ldots, \check{x}_m \in \check{\mathcal{D}}_{\tau_m}$. The bijectivity of $\natural_\tau$ follows trivially.
4. Let $\natural_{\tau_i}$ be defined for $\mathcal{D}^{"\tau_1"}, \ldots, \mathcal{D}^{"\tau_m"}$. We are going to define a function $\natural_\tau$ (for order of τ is less than n) with $\tau = (\tau_1 \times \cdots \times \tau_m \to o)$ for $\mathcal{D}^{"\tau"}$. For all $x \in \mathcal{D}^{"\tau"}$ $\natural_\tau(x)$ is defined as the predicate $\natural_\tau(x)(\check{x}_1, \ldots, \check{x}_m) := \mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(x, \natural_{\tau_1}^{-1}(\check{x}_1), \ldots, \natural_{\tau_m}^{-1}(\check{x}_m))$ for all $\check{x}_1 \in \check{\mathcal{D}}_{\tau_1}, \ldots, \check{x}_m \in \check{\mathcal{D}}_{\tau_m}$. Analogously to case 2 we get the bijectivity of $\natural_\tau$ by the corresponding formula in $\%AXIOMS$.

$\natural$ is the polymorphic mapping defined by all the individual $\natural_\tau$. Now we are going to show that if $\mathcal{M}$ is a model of $\hat{\Theta}(\varphi)$, that is, for all assignments ξ we have $\mathcal{V}_\xi^{\mathcal{M}}(\hat{\Theta}(\varphi)) = \mathtt{T}$, we have $\check{\mathcal{M}}$ is a model of φ, that is, for all assignments $\check{\xi}$ we have $\mathcal{V}_{\check{\xi}}^{\check{\mathcal{M}}}(\varphi) = \mathtt{T}$ with $\check{\mathcal{M}} = \langle\{\check{\mathcal{D}}_\tau\}_\tau, \check{\mathcal{J}}\rangle$. $\check{\mathcal{J}}$ is defined as $\natural \circ \mathcal{J} \circ \hat{\Theta}'$. The assignments $\check{\xi}$ are defined as $\natural \circ \xi \circ \hat{\Theta}'$. Because $\natural$ and $\hat{\Theta}'$ are bijective, we get all assignments this way.

We now prove by induction on the construction of terms, that for all terms: $\mathcal{V}_{\check{\xi}}^{\check{\mathcal{M}}} = \natural \circ \mathcal{V}_\xi^{\mathcal{M}} \circ \hat{\Theta}'$.

*Since we cannot achieve bijectivity from $\mathcal{D}^{"\tau"}$ to $\mathcal{F}(\check{\mathcal{D}}_{\tau_1}, \ldots, \check{\mathcal{D}}_{\tau_m}; \check{\mathcal{D}}_\sigma)$ we do not get strong completeness.

T1 For all variables x_τ, $\mathcal{V}_\xi^{\check{\mathcal{M}}}(x_\tau) = \check{\xi}(x_\tau) = \natural(\xi(\hat{\Theta}'(x_\tau))) = \natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(x_\tau)$.

T2 For all constants c_τ, $\mathcal{V}_\xi^{\check{\mathcal{M}}}(c_\tau) = \check{\mathcal{J}}(c_\tau) = \natural(\mathcal{J}(\hat{\Theta}'(c_\tau))) = \natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(c_\tau)$.

T3 For all composed terms beginning with an m-ary function term f we have:
$\mathcal{V}_\xi^{\check{\mathcal{M}}}(f_\tau(t_1,\ldots,t_m)) = \mathcal{V}_\xi^{\check{\mathcal{M}}}(f)(\mathcal{V}_\xi^{\check{\mathcal{M}}}(t_1),\ldots,\mathcal{V}_\xi^{\check{\mathcal{M}}}(t_m)) \overset{\text{Ind.hyp}}{=}$
$\natural(\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(f))(\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_1),\ldots,\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_m)) \overset{\text{def } \natural \text{ case 2.}}{=}$
$\natural[\mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(f),\natural^{-1}\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_1),\ldots,\natural^{-1}\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_m))] \overset{\text{def } \mathcal{V}_\xi^{\mathcal{M}}}{=}$
$\natural[\mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"}(\hat{\Theta}'(f),\hat{\Theta}'(t_1),\ldots,\hat{\Theta}'(t_m)))] \overset{\text{def } \hat{\Theta}'}{=} \natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(f(t_1,\ldots,t_m))$.

We now prove inductively, that for all formulae: $\mathcal{V}_\xi^{\check{\mathcal{M}}} = \mathcal{V}_\xi^{\mathcal{M}} \circ \hat{\Theta}'$.

F1 For an atomic formula that starts with a predicate constant p of order n we have:
$\mathcal{V}_\xi^{\check{\mathcal{M}}}(p_\tau(t_1,\ldots,t_m)) = \mathcal{V}_\xi^{\check{\mathcal{M}}}(p)(\mathcal{V}_\xi^{\check{\mathcal{M}}}(t_1),\ldots,\mathcal{V}_\xi^{\check{\mathcal{M}}}(t_m)) \overset{\text{Ind.hyp}}{=}$
$\natural[\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(p)](\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_1),\ldots,\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_m)) \overset{\text{def } \natural \text{ case 3.}}{=}$
$\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(p)(\natural^{-1}\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_1),\ldots,\natural^{-1}\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_m)) \overset{\text{def } \mathcal{V}_\xi^{\mathcal{M}}}{=} \mathcal{V}_\xi^{\mathcal{M}}(\hat{\Theta}'(p)(\hat{\Theta}'(t_1),\ldots,\hat{\Theta}'(t_m))) \overset{\text{def } \hat{\Theta}'}{=}$
$\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(p(t_1,\ldots,t_m))$.

F2 For an atomic formula that starts with a predicate term p of order less than n we have:
$\mathcal{V}_\xi^{\check{\mathcal{M}}}(p_\tau(t_1,\ldots,t_m)) = \mathcal{V}_\xi^{\check{\mathcal{M}}}(p)(\mathcal{V}_\xi^{\check{\mathcal{M}}}(t_1),\ldots,\mathcal{V}_\xi^{\check{\mathcal{M}}}(t_m)) \overset{\text{Ind.hyp}}{=}$
$\natural[\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(p)](\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_1),\ldots,\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_m)) \overset{\text{def } \natural \text{ case 4.}}{=}$
$\mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(p),\natural^{-1}\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_1),\ldots,\natural^{-1}\natural\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(t_m)) \overset{\text{def } \mathcal{V}_\xi^{\mathcal{M}}}{=}$
$\mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"}(\hat{\Theta}'(p),\hat{\Theta}'(t_1),\ldots,\hat{\Theta}'(t_m))) \overset{\text{def } \hat{\Theta}'}{=} \mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(p(t_1,\ldots,t_m))$.

F3 For a conjunction we have:
$\mathcal{V}_\xi^{\check{\mathcal{M}}}(\varphi_1 \wedge \varphi_2) = \mathcal{V}_\xi^{\check{\mathcal{M}}}(\varphi_1) \wedge \mathcal{V}_\xi^{\check{\mathcal{M}}}(\varphi_2) \overset{\text{Ind.hyp}}{=} \mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(\varphi_1) \wedge \mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(\varphi_2) = \mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(\varphi_1 \wedge \varphi_2)$.

F4 For a negation we have:
$\mathcal{V}_\xi^{\check{\mathcal{M}}}(\neg\varphi) = \neg\mathcal{V}_\xi^{\check{\mathcal{M}}}(\varphi) \overset{\text{Ind.hyp}}{=} \neg\mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(\varphi) = \mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(\neg\varphi)$.

F5 For a quantification we have:
$\mathcal{V}_\xi^{\check{\mathcal{M}}}(\forall x_\tau\varphi) = \forall d \in \check{\mathcal{D}}_\tau \mathcal{V}_{\check{\xi}[x\leftarrow d]}^{\check{\mathcal{M}}}(\varphi) \overset{\text{Ind.hyp}}{=} \forall d \in \check{\mathcal{D}}_\tau \mathcal{V}_{\xi[\hat{\Theta}'(x)\leftarrow\natural^{-1}(d)]}^{\mathcal{M}}\hat{\Theta}'(\varphi) =$
$\forall d \in \mathcal{D}^{"\tau"} \mathcal{V}_{\xi[\hat{\Theta}'(x)\leftarrow d]}^{\mathcal{M}}\hat{\Theta}'(\varphi) = \mathcal{V}_\xi^{\mathcal{M}}\hat{\Theta}'(\forall x\varphi)$.
Here we use that $\check{\xi} = \natural\circ\xi\circ\hat{\Theta}'$ for all assignments and hence $\check{\xi}[x_\tau \leftarrow d] = \natural\circ\xi[\hat{\Theta}'(x_\tau) \leftarrow \natural^{-1}(d)]\circ\hat{\Theta}'$.
(In the strict sense we had to do the induction proof for assigments $\check{\xi}[x \leftarrow d]$ as well.)

Summarizing we have: if $\mathcal{V}_\xi^{\mathcal{M}}(\hat{\Theta}(\varphi)) = \mathtt{T}$ then $\mathcal{V}_\xi^{\mathcal{M}}(\hat{\Theta}'(\varphi)) = \mathtt{T}$ then $\mathcal{V}_\xi^{\check{\mathcal{M}}}(\varphi) = \mathtt{T}$. ■

6.8 Remark: For $n > 1$ there is no sound morphism Θ from $\mathcal{L}^n$ to $\mathcal{L}^1_{sort}$ which is strongly complete. If there were such a morphism it would provide a complete calculus for $\mathcal{L}^n$ which is impossible because of GÖDEL'S incompleteness theorem.

6.9 Remark: As already noticed in remark 5.5, $\hat{\Theta}'^{-1}$ provides a calculus for $\mathcal{L}^n$. If we add rules that enforce that function symbols and predicate symbols are equal if they agree in all arguments, we can transform every sound and complete first order calculus of $\mathcal{L}^1_{sort}$ by $\hat{\Theta}$ to a sound and weakly complete calculus for $\mathcal{L}^n$. We can execute the proof in $\mathcal{L}^1_{sort}$ and then lift it to a proof in $\mathcal{L}^n$.

6.10 Remark: One might wonder why we proposed a sufficient criterion for the soundness of translations, when we have a translation that is sound and complete and hence could be used always. However in a concrete situation it can be better not to translate into the full sound and complete formulae,

because the search space may become too big. It would not be a good idea to add the extensionality axioms if they are not really needed. In addition we can prevent instantiation if we translate certain constants not by an *apply* or if we use different *apply* functions or predicates although we could use the same. On the other hand the completeness result guarantees that we can find a translation at all. Which one we choose may be very important for the theorem prover to find a proof.

7 EQUALITY

In this chapter we discuss a possible extension from $\mathcal{L}^n$ to $\mathcal{L}^n_{\equiv}$, by extending the soundness criterion and an extension of the morphisms $\hat{\Theta}_n$ to morphisms $\hat{\Theta}_{\equiv,n}$, which are mappings form $\mathcal{L}^n_{\equiv}$ to $\mathcal{L}^1_{sort}$. As usual we fix n and drop the corresponding index. We show that $\hat{\Theta}_{\equiv}$ is sound and weakly complete. In the following we write $\bar{\tau}$ for $(\tau \times \tau \to o)$.

7.1 Definition (Equality Quasi-Homomorphism): The inductive case of the condition for terms has to be replaced in definition 4.4 by: if $f(t_1,\ldots,t_m)$ is a term of $\mathcal{F}^1(\mathcal{S}_1)$, then $\Theta(f(t_1,\ldots,t_m)) = \vartheta(\Theta(f),\Theta(t_1),\ldots,\Theta(t_m))$ with
$\vartheta(a,a_1,\ldots,a_m) = \begin{cases} a(a_1,\ldots,a_m) & \text{or} \\ \alpha_a(a,a_1,\ldots,a_m) \end{cases}$
The α have to be chosen appropriately out of $\mathcal{S}_2$, especially they have to be *new*, that is, there must be no element $\alpha' \in \mathcal{S}_1$ so that $\alpha_a = \Theta(\alpha')$. The case which is chosen can depend only on the a not on the $a_1,\ldots,a_m$. *If the first case is chosen for equality, equality must be mapped onto equality.*

7.2 Theorem: If Θ is an injective equality quasi-homomorphism from $\mathcal{L}^n_{\equiv}(\mathcal{S})$ to $\mathcal{L}^1_{sort}(\mathcal{S}')$, then Θ is weakly sound.

Proof: The proof is analogous to the proof of theorem 5.1. We only add the following cases to the proof (analogously to the cases T3, T4):

- For an atomic formula with the equality symbol as top symbol that is mapped onto the equality predicate:
  $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(t_1 \equiv_{\bar{\tau}} t_2)) := \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(t_1) \equiv^{(\text{"}\tau\text{"},\text{"}\tau\text{"})} \Theta(t_2)) = \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(t_1)) \equiv_{\mathcal{D}^{\text{"}\tau\text{"}}} \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(t_2)) \overset{\text{Ind.hyp}}{=}$
  $\mathcal{V}^{\mathcal{M}}_{\xi}(t_1) \equiv_{\mathcal{D}_\tau} \mathcal{V}^{\mathcal{M}}_{\xi}(t_2) = \mathcal{V}^{\mathcal{M}}_{\xi}(t_1 \equiv_{\bar{\tau}} t_2)$
- For an atomic formula with an equality symbol as top symbol that is not mapped onto the equality predicate we have:
  $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(t_1 \equiv_{\bar{\tau}} t_2)) := \mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\alpha^{\text{"}\bar{\tau}\text{"}}(\Theta(\equiv_{\bar{\tau}}),\Theta(t_1),\Theta(t_2))) =$
  $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\alpha^{\text{"}\bar{\tau}\text{"}})(\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(\equiv_{\bar{\tau}})),\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(t_1)),\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\Theta(t_2))) \overset{\text{Ind.hyp}}{=}$
  $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\alpha^{\text{"}\bar{\tau}\text{"}})(\mathcal{V}^{\mathcal{M}}_{\xi}(\equiv_{\bar{\tau}}),\mathcal{V}^{\mathcal{M}}_{\xi}(t_1),\mathcal{V}^{\mathcal{M}}_{\xi}(t_2)) \overset{\text{def}}{=} \mathcal{V}^{\mathcal{M}}_{\xi}(t_1) \equiv_{\mathcal{D}_\tau} \mathcal{V}^{\mathcal{M}}_{\xi}(t_2) =$
  $\mathcal{V}^{\mathcal{M}}_{\xi}(t_1 \equiv_{\bar{\tau}} t_2)$ ∎

7.3 Example: We shall use example 5.3, however in a formulation with equality and translate it then in the usual way. (In order to show that both representations are *equivalent* it would be necessary to show that there is a sound and complete morphism that maps them to one another.) We drop the type information for readability, *group* is of type $(\iota \times (\iota \times \iota \to \iota) \to o)$, G of type ι, $+$ of type $(\iota \times \iota \to \iota)$, $-$ of type $(\iota \to \iota)$, and so on. Also for readability we sometimes use infix notation. In the target the sorts are ("ι", "$(\iota \times \iota \to \iota)$") for *group*, and so on. A group can be defined as follows:

1. $\forall G,+\ \ group(G,+) \iff associative(G,+) \wedge$
   $\exists 0\ \ (0 \in G \wedge neutral\_element(G,+,0) \wedge$
   $\exists -\ \ inverse(G,+,0,-))$
2. $\forall G,+\ \ associative(G,+) \iff \forall x,y,z\ \ x,y,z \in G \Longrightarrow (x+y)+z \equiv x+(y+z)$
3. $\forall G,+,0\ \ neutral\_element(G,+,0) \iff \forall x\ \ x \in G \Longrightarrow x+0 \equiv x \ \wedge\ 0+x \equiv x$

4. $\forall G,+,0,-\ \ inverse(G,+,0,-) \iff \forall x\ \ x \in G \Longrightarrow x+(-x) \equiv 0\ \wedge\ (-x)+x \equiv 0$

This formula set is a subset of $\mathcal{L}^3_{\equiv}$. Now we give a translation into a formulae set of $\mathcal{L}^1_{sort}$. The signatures are obvious, hence ommitted. The translation is sound, because it is an injective equality quasi-homomorphism.

1. $\forall G,+\ \ group(G,+) \iff associative(G,+) \wedge$
$\exists 0\ (0 \in G\ \wedge\ neutral\_element(G,+,0) \wedge$
$\exists -\ \ inverse(G,+,0,-))$

2. $\forall G,+\ \ associative(G,+) \iff \forall x,y,z\ \ x \in G \wedge\ y \in G \wedge\ z \in G \Longrightarrow$
$\boldsymbol{\alpha}^{"(\iota\times\iota\to\iota)"}(+,\boldsymbol{\alpha}^{"(\iota\times\iota\to\iota)"}(+,x,y),z) \equiv$
$\boldsymbol{\alpha}^{"(\iota\times\iota\to\iota)"}(+,x,\boldsymbol{\alpha}^{"(\iota\times\iota\to\iota)"}(+,y,z))$

3. $\forall G,+,0\ \ neutral\_element(G,+,0) \iff \forall x\ \ x \in G \Longrightarrow$
$\boldsymbol{\alpha}^{"(\iota\times\iota\to\iota)"}(+,x,0) \equiv x \wedge\ \boldsymbol{\alpha}^{"(\iota\times\iota\to\iota)"}(+,0,x) \equiv x$

4. $\forall G,+,0,-\ \ inverse(G,+,0,-) \iff \forall x\ \ x \in G \Longrightarrow$
$\boldsymbol{\alpha}^{"(\iota\times\iota\to\iota)"}(+,x,\boldsymbol{\alpha}^{"(\iota\to\iota)"}(-,x)) \equiv 0 \wedge$
$\boldsymbol{\alpha}^{"(\iota\times\iota\to\iota)"}(+,\boldsymbol{\alpha}^{"(\iota\to\iota)"}(-,x),x) \equiv 0$

7.4 Definition (Standard Translation $\hat{\Theta}_{\equiv}$):

- At first we define the mapping on the signature. We proceed as in definition 6.1, but add for each $\equiv_\tau$ in $\mathcal{S}_\tau$ of order less than n an object-constant symbol $\hat{\equiv}^{"\tau"}$ in $\mathcal{S}^{"\tau"}$. We cannot name it $\equiv^{("\tau","\tau")}$ because this is already defined as a binary predicate symbol.
- Like above on formula sets we define $\hat{\Theta}_{\equiv}(\Gamma) := \hat{\Theta}'_{\equiv}(\Gamma) \cup \%AXIOMS_{\equiv}$.
- The inductive definition of $\hat{\Theta}'_{\equiv}(\Gamma)$ is the definition of $\hat{\Theta}'(\Gamma)$ in definition 6.1 plus
$\hat{\Theta}'_{\equiv}(\equiv_{\bar\tau}) = \equiv^{("\tau","\tau")}$ for order of $\bar\tau$ equal to n and $\hat{\Theta}'_{\equiv}(\equiv_{\bar\tau}) = \hat{\equiv}^{"\bar\tau"}$ for order of $\bar\tau$ less than n.
In addition we have:
If t_1 and t_2 are terms of type τ with $\tau \neq o$ and order of $\bar\tau$ is equal to n, then $\hat{\Theta}'_{\equiv}(t_1 \equiv_{\bar\tau} t_2) = \hat{\Theta}'_{\equiv}(t_1) \equiv^{("\tau","\tau")} \hat{\Theta}'_{\equiv}(t_2)$. This term is well-sorted, because $\hat{\Theta}_{\equiv}(t_i)$ are both of sort $"\tau"$.
If t_1 and t_2 are terms of type τ with $\tau \neq o$ and order of $\bar\tau$ less than n, then $\hat{\Theta}'_{\equiv}(t_1 \equiv_{\bar\tau} t_2) = \boldsymbol{\alpha}^{"(\tau\times\tau\to o)"}(\hat{\equiv}^{"\bar\tau"},\hat{\Theta}'_{\equiv}(t_1),\hat{\Theta}'_{\equiv}(t_2))$. This term is again well-sorted, because $\hat{\Theta}_{\equiv}(t_i)$ are both of sort $"\tau"$ and $\hat{\equiv}^{"\bar\tau"}$ is of type $"(\tau \times \tau \to o)"$.
- $\%AXIOMS_{\equiv}$ is defined as $\%AXIOMS$ plus the set of all formulae (with order of $\bar\tau$ less than n):
$\forall x^{"\tau"}\forall y^{"\tau"}\boldsymbol{\alpha}^{"\bar\tau"}(\hat{\equiv}^{"\bar\tau"},x,y) \Longrightarrow x \equiv^{("\tau","\tau")} y$.

7.5 Theorem: $\hat{\Theta}_{\equiv}$ is weakly sound.

Proof: As above we have that $\hat{\Theta}'_{\equiv}$ is sound because it is an injective equality quasi-homomorphism. We still have to show that every formula in $\%AXIOMS_{\equiv}$ is satisfied, that is, it remains to be shown:
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat\xi}(\forall x^{"\tau"}\forall y^{"\tau"}\boldsymbol{\alpha}^{"\bar\tau"}(\hat{\equiv}^{"\bar\tau"},x,y) \Longrightarrow x \equiv^{"\bar\tau"} y) = \mathrm{T}$
Therefore it is sufficient to show that for all $X,Y \in \mathcal{D}^{"\tau"}$
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat\xi[x,y\leftarrow X,Y]}(\boldsymbol{\alpha}^{"\bar\tau"}(\hat{\equiv}^{"\bar\tau"},x,y) \Longrightarrow x \equiv^{"\bar\tau"} y) = \mathrm{T}$.
By the definitions of $\boldsymbol{\alpha}^{"\bar\tau"}$ and $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat\xi[x,y\leftarrow X,Y]}$ that is equivalent to
$X \equiv_{\mathcal{D}_\tau} Y \Longrightarrow X \equiv_{\mathcal{D}_\tau} Y$, which is obviously true. ∎

7.6 Theorem: $\hat{\Theta}_{\equiv}$ is weakly complete.

Proof: In the completeness theorem 6.7 we have to add to the proof:

- For order equal to n:
$\mathcal{V}^{\hat{\mathcal{M}}}_{\hat\xi}((t_1 \equiv_{\bar\tau} t_2)) := \mathcal{V}^{\hat{\mathcal{M}}}_{\hat\xi}(t_1) \equiv_{\mathcal{D}_\tau} \mathcal{V}^{\hat{\mathcal{M}}}_{\hat\xi}(t_2) \overset{\text{Ind.hyp}}{=} \natural\mathcal{V}^{\mathcal{M}}_{\hat\xi}\hat{\Theta}'(t_1) \equiv_{\mathcal{D}_\tau} \natural\mathcal{V}^{\mathcal{M}}_{\hat\xi}\hat{\Theta}'(t_2) \overset{\natural\ \text{bij}}{=}$
$\mathcal{V}^{\mathcal{M}}_{\hat\xi}\hat{\Theta}'(t_1) \equiv_{\mathcal{D}^{"\tau"}} \mathcal{V}^{\mathcal{M}}_{\hat\xi}\hat{\Theta}'(t_2) = \mathcal{V}^{\mathcal{M}}_{\hat\xi}(\hat{\Theta}'(t_1) \equiv^{("\tau","\tau")} \hat{\Theta}'(t_2) = \mathcal{V}^{\mathcal{M}}_{\hat\xi}\hat{\Theta}'(t_1 \equiv_{\bar\tau} t_2)$.

- For the equalities of order less than n we use at first the additional axioms in $\%AXIOMS_{\equiv}$:
  $\forall x^{"\tau"} \forall y^{"\tau"} \alpha^{"\bar{\tau}"}(\hat{\equiv}^{"\bar{\tau}"}, x, y) \Longrightarrow x \equiv^{("\tau","\tau")} y$.
  Hence we have in a model for all X, Y in $\mathcal{D}^{"\tau"}$:
  $\mathcal{V}^{\mathcal{M}}_{\xi[x,y \leftarrow X,Y]}(\alpha^{"\bar{\tau}"}(\hat{\equiv}^{"\bar{\tau}"}, x, y)) \Longrightarrow X \equiv_{\mathcal{D}^{"\tau"}} Y$
  Since the direction $\Longleftarrow$ is trivially satisfied, we have:
  $\mathcal{V}^{\mathcal{M}}_{\xi[x,y \leftarrow X,Y]}(\alpha^{"\tau"}(\hat{\equiv}^{"\bar{\tau}"}, x, y)) = X \equiv_{\mathcal{D}^{"\tau"}} Y \qquad (*)$
  Now we can prove:
  $\mathcal{V}^{\check{\mathcal{M}}}_{\xi}(t_1 \equiv_{\bar{\tau}} t_2) = \mathcal{V}^{\check{\mathcal{M}}}_{\xi} t_1 \equiv_{\mathcal{D}_\tau} \mathcal{V}^{\check{\mathcal{M}}}_{\xi} t_2 = \natural \mathcal{V}^{\mathcal{M}}_{\xi} \hat{\Theta}'_{\equiv}(t_1) \equiv_{\mathcal{D}_\tau} \natural \mathcal{V}^{\mathcal{M}}_{\xi} \hat{\Theta}'_{\equiv}(t_2) \overset{\natural \text{ bij}}{=}$
  $\mathcal{V}^{\mathcal{M}}_{\xi} \hat{\Theta}'_{\equiv}(t_1) \equiv_{\mathcal{D}^{"\tau"}} \mathcal{V}^{\mathcal{M}}_{\xi} \hat{\Theta}'_{\equiv}(t_2) \overset{(*)}{=} \mathcal{V}^{\mathcal{M}}_{\xi[x,y \leftarrow \mathcal{V}^{\mathcal{M}}_{\xi} \hat{\Theta}'_{\equiv}(t_1), \mathcal{V}^{\mathcal{M}}_{\xi} \hat{\Theta}'_{\equiv}(t_2)]}(\alpha^{"\bar{\tau}"}(\hat{\equiv}^{"\bar{\tau}"}, x, y)) =$
  $\mathcal{V}^{\mathcal{M}}_{\xi}(\alpha^{"\bar{\tau}"}(\hat{\equiv}^{"\bar{\tau}"}, \hat{\Theta}'_{\equiv}(t_1), \hat{\Theta}'_{\equiv}(t_2))) = \mathcal{V}^{\mathcal{M}}_{\xi}(\alpha^{"\bar{\tau}"}(\hat{\Theta}'_{\equiv}(\equiv_{\bar{\tau}}), \hat{\Theta}'_{\equiv}(t_1), \hat{\Theta}'_{\equiv}(t_2))) =$
  $\mathcal{V}^{\mathcal{M}}_{\xi} \hat{\Theta}'_{\equiv}(t_1 \equiv_{\bar{\tau}} t_2)$ ∎

7.7 Remark: We do not translate $\equiv_{\bar{\tau}}$ immediately to $\equiv^{("\tau","\tau")}$, because then it could not become the argument of higher order predicates, we would also lose completeness. Consider the case of the following induction schema:
$\forall P_{(\iota \times \iota \to o)}(P(0,0) \wedge (\forall n \;\; P(n,0) \Longrightarrow P(s(n),0)) \wedge (\forall n, m \;\; P(n,m) \Longrightarrow P(n, s(m)))$
$\Longrightarrow \forall n, m \;\; P(n,m))$,
where in addition we have the formulae $0 \equiv 0$, $\forall n \;\; n \equiv 0 \Longrightarrow s(n) \equiv 0$, and $\forall n, m \;\; n \equiv m \Longrightarrow n \equiv s(m)$. If we want to prove $\forall n, m \;\; n \equiv m$ we have to instantiate the predicate variable P in the induction schema by the equality predicate. But if we translate P by an object variable and $\equiv$ by a predicate constant we cannot instantiate in the first order target formulation P by $\equiv$.

7.8 Example: We translate the examples 5.3 and 7.3 from above. Using $\hat{\Theta}'_{\equiv,3}$ this is translated to:

1. $\forall G, + \;\; \mathit{group}(G,+) \Longleftrightarrow \mathit{associative}(G,+) \wedge$
   $\exists 0 \;\; (0 \in G \wedge \mathit{neutral\_element}(G,+,0) \wedge$
   $\exists - \;\; \mathit{inverse}(G,+,0,-))$
2. $\forall G, + \;\; \mathit{associative}(G,+) \Longleftrightarrow$
   $\forall x, y, z \;\; \alpha^{"(\iota \times \iota \to o)"}(\in, x, G) \wedge \alpha^{"(\iota \times \iota \to o)"}(\in, y, G) \wedge \alpha^{"(\iota \times \iota \to o)"}(\in, z, G) \Longrightarrow$
   $\alpha^{"(\iota \times \iota \to o)"}(\hat{\in}, \alpha^{"(\iota \times \iota \to \iota)"}(+, \alpha^{"(\iota \times \iota \to \iota)"}(+, x, y), z),$
   $\alpha^{"(\iota \times \iota \to \iota)"}(+, x, \alpha^{"(\iota \times \iota \to \iota)"}(+, y, z)))$
3. $\forall G, +, 0 \;\; \mathit{neutral\_element}(G,+,0) \Longleftrightarrow$
   $\forall x \;\; \alpha^{"(\iota \times \iota \to o)"}(\in, x, G) \Longrightarrow$
   $\alpha^{"(\iota \times \iota \to o)"}(\hat{\in}, \alpha^{"(\iota \times \iota \to \iota)"}(+, x, 0), x) \wedge \alpha^{"(\iota \times \iota \to o)"}(\hat{\in}, \alpha^{"(\iota \times \iota \to \iota)"}(+, 0, x), x)$
4. $\forall G, +, 0, - \;\; \mathit{inverse}(G,+,0,-) \Longleftrightarrow$
   $\forall x \;\; \alpha^{"(\iota \times \iota \to o)"}(\in, x, G) \Longrightarrow$
   $\alpha^{"(\iota \times \iota \to o)"}(\hat{\in}, \alpha^{"(\iota \times \iota \to \iota)"}(+, x, \alpha^{"(\iota \to \iota)"}(-, x)), 0) \wedge$
   $\alpha^{"(\iota \times \iota \to o)"}(\hat{\in}, \alpha^{"(\iota \times \iota \to \iota)"}(+, \alpha^{"(\iota \to \iota)"}(-, x), x), 0)$

Of course this translation is more complicate than that of example 7.3.

8 Summary and Open Problems

In the sections above we introduced the basic machinery for translating higher order formulae to first order logic. We introduced a sufficient criterion for the soundness of such a translation, namely that it has to be an injective quasi-homomorphism. Then we gave a complete translation for the restricted higher order language. In the last section we generalized these results to logics with equality.

An interesting and useful generalization would be to a higher order *sorted* logic. Then the first order logic should of course have a sort structure at least as powerful as that of the higher order source logic. The results should be transferable although the formal treatment can become strenuous.

ACKNOLEWDGEMENT

I like to thank AXEL PRÄCKLEIN for many discussions and thorough reading of a draft and JÖRG SIEKMANN for his advice that resulted in numerous improvements.

REFERENCES

[1] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof.* Academic Press, Orlando, Florida, USA, 1986.

[2] Peter B. Andrews, Sunil Issar, Dan Nesmith, and Frank Pfenning. The TPS theorem proving system. In M.E. Stickel, editor, *Proc. of the 10th CADE*, pages 641–642, Kaiserslautern, Germany, July 1990. Springer Verlag, Berlin, Germany. LNAI 449.

[3] Johan van Benthem and Kees Doets. *Higher Order Logic*, volume I: Elements of Classical Logic of *Handbook of Philosophical Logic, D. Gabbay, F. Guenthner, editors*, chapter I.4, pages 275–329. D.Reidel Publishing Company, Dodrecht, Netherlands, 1983.

[4] Nicolas Bourbaki. *Théorie des ensembles.* Éléments de mathématique, Fascicule 1. Hermann, Paris, France, 1954.

[5] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, **5**:56–68, 1940.

[6] Peter Deussen. *Halbgruppen und Automaten*, volume 99 of *Heidelberger Taschenbücher.* Springer-Verlag, Berlin, Germany, 1971.

[7] Adolf Abraham Fraenkel. Zu den Grundlagen der Cantor-Zermeloschen Mengenlehre. *Mathematische Annalen*, **86**:230–237, 1922.

[8] Kurt Gödel. *The Consistency of the Axiom of Choice and of the Generalized Continuum-Hypothesis with the Axioms of Set Theory*, volume 3 of *Annals of Mathematics Studies.* Princeton University Press, Princeton, New Jersey, eighth printing 1970, 1940.

[9] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, **15**:81–91, 1950.

[10] Andrzej Mostowski. An undecidable arithmetical statement. *Fundamenta Mathematicae*, **36**:143–164, 1949.

[11] Arnold Oberschelp. Untersuchungen zur mehrsortigen Quantorenlogik. *Mathematische Annalen*, **145**:297–333, 1962.

[12] Hans Jürgen Ohlbach. Context logic. SEKI Report SR-89-08, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1989.

[13] Karl Mark G Raph. The Markgraf Karl Refutation Procedure. Technical Report Memo-SEKI-MK-84-01, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, January 1984.

[14] Bertrand Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, XXX:222–262, 1908.

[15] Manfred Schmidt-Schauß. *Computational Aspects of an Order-Sorted Logic with Term Declarations*, volume 395 of *Lecture Notes in Artificial Intelligence.* Springer Verlag, Berlin, Germany, 1989.

[16] Christoph Walther. Ein mehrsortiger Resolutionskalkül mit Paramodulation. Interner Bericht 35/82, Fakultät für Informatik, Universität Karlsruhe, Karlsruhe, Germany, 1982.

[17] Ernst Zermelo. Untersuchungen über die Grundlagen der Mengenlehre. I. *Mathematische Annalen*, **86**:230–237, 1922.

Appendix: An Example of a Proof

In this appendix we give an example for a theorem and a proof that was actually generated by the MKRP-prover [13]. The example is part of theorem (4.12) in [6, p.41]. The theorem can be formulated as follows:

Theorem: Let S be a set, let ρ_1 and ρ_2 be equivalence relations, let Φ be a surjective mapping from S/ρ_1 onto S/ρ_2, and let $\natural_i$ be the canonical projections from S to S/ρ_i, such that $\Phi \circ \natural_1 = \natural_2$, then $\rho_1 \subseteq \rho_2$.

A possible axiomatization is:

1. $set(S)$
2. Definition of a rest class modulo an equivalence relation:
   $\forall x_\iota \ \forall \rho_\iota \ \ set(x) \wedge equivalence.relation(\rho, x) \Longrightarrow set(x/\rho)$
3. Definition of a canonical.projection:
   $\forall \rho_\iota \ \ equivalence.relation(\rho, s) \Longrightarrow mapping(canonical.projection(\rho, S), S, S/\rho)$
4. Theorem 4-11-1:
   $\forall U_\iota, V_\iota \ \forall \varphi^1_{(\iota\to\iota)}, \varphi^2_{(\iota\to\iota)} \ \ set(U) \wedge set(V) \wedge mapping(\varphi^1, S, U) \wedge mapping(\varphi^2, S, V) \Longrightarrow$
   $\forall \Phi_{(\iota\to\iota)} \ \ mapping(\Phi, U, V) \wedge (\forall t_\iota \ \ t \in S \Longrightarrow \Phi(\varphi^1(t)) = \varphi^2(t)) \Longrightarrow$
   $induced.equivalence.relation(\varphi^1, S, U) \subseteq induced.equivalence.relation(\varphi^2, S, U)$
5. Part of Definition of surjectivity:
   $\forall \varphi_{(\iota\to\iota)} \ \ \forall U_\iota, V_\iota \ \ set(U) \wedge set(V) \Longrightarrow (surjective(\varphi, U, V) \Longrightarrow mapping(\varphi, U, V))$
6. Lemma:
   $\forall \rho_\iota \ \ equivalence.relation(\rho, S) \Longrightarrow$
   $induced.equivalence.relation(canonical.projection(\rho, S), S, S/\rho) = \rho$
7. Theorem to prove: THEOREM 4-12-1:
   $\forall \rho^1_\iota, \rho^2_\iota \ \ equivalence.relation(\rho^1, S) \ \wedge \ equivalence.relation(\rho^2, S) \Longrightarrow$
   $((\forall \Phi_{(\iota\to\iota)} \ \ surjective(\Phi, S/\rho^1, S/\rho^2) \wedge$
   $(\forall a_\iota \ \ a \in S \Longrightarrow \Phi(canoncical.projection(\rho^1, S)(a)) =$
   $canonical.projection(\rho^2, S)(a))) \Longrightarrow$
   $\rho^1 \subseteq \rho^2)$.

Translated by $\hat{\Theta}'_3$ into the Markgraf Karl notation (with ι written as `I`, o as `0`, $\times$ as `X`, and $\to$ as `T`):

```
Formulae given to the editor
============================


Axioms:   SORT I,ITI,ITO,IXITI,IXITO,[ITI]XIXITI,IXIT[ITI] : ANY
          TYPE S : I
          TYPE SET : ITO
          TYPE SUBSET : IXITO
          TYPE CANONICAL.PROJECTION : IXIT[ITI]
          TYPE MODULO : IXITI
          TYPE IN : IXITO
          TYPE INDUCED.EQUIVALENCE.RELATION : [ITI]XIXITI
          TYPE EQUIVALENCE.RELATION : IXITO
          TYPE MAPPING (ITI I I)
          TYPE SURJECTIVE (ITI I I)
          TYPE APPLY-IXITI (IXITI I I) : I
          TYPE APPLY-ITI (ITI I) : I
          TYPE APPLY-IXITO (IXITO I I)
          TYPE APPLY-[ITI]XIXITI ([ITI]XIXITI ITI I I) : I
          TYPE APPLY-IXIT[ITI] (IXIT[ITI] I I) : ITI
          TYPE APPLY-ITO (ITO I)
          APPLY-ITO (SET S)
          * PART OF DEFINITION OF REST CLASSES MODULO AN EQUIVALENCE RELATION *
          ALL X : I ALL RHO : I APPLY-ITO (SET X) AND APPLY-IXITO (EQUIVALENCE.RELATION RHO X)
                       IMPL APPLY-ITO (SET APPLY-IXITI (MODULO X RHO))
```

```
        * DEFINITION IN THEOREM 4.1 *
        ALL RHO : I APPLY-IXITO (EQUIVALENCE.RELATION RHO S)
                    IMPL MAPPING (APPLY-IXIT[ITI] (CANONICAL.PROJECTION RHO S)
                                  S
                                  APPLY-IXITI (MODULO S RHO))
        * THEOREM 4-11-1 *
        ALL U,V: I ALL PHI1,PHI2 : ITI
          APPLY-ITO (SET U) AND APPLY-ITO (SET V) AND MAPPING (PHI1 S U) AND MAPPING (PHI2 S V)
          IMPL (ALL PPHI: ITI MAPPING (PPHI U V)
                     AND (ALL T : I APPLY-IXITO (IN T S)
                                    IMPL APPLY-ITI (PPHI APPLY-ITI (PHI1 T)) =
                                         APPLY-ITI (PHI2 T))
                     IMPL APPLY-IXITO (SUBSET
                                         APPLY-[ITI]XIXITI (INDUCED.EQUIVALENCE.RELATION PHI1 S U)
                                         APPLY-[ITI]XIXITI (INDUCED.EQUIVALENCE.RELATION PHI2 S V)))
        * PARTIAL DEFINITION OF SURJECTIVITY *
        ALL PHI : ITI ALL U,V : I APPLY-ITO (SET U) AND APPLY-ITO (SET V)
                                  IMPL (SURJECTIVE (PHI U V) IMPL MAPPING (PHI U V))
        * LEMMA *
        ALL RHO : I APPLY-IXITO (EQUIVALENCE.RELATION RHO S)
                                IMPL APPLY-[ITI]XIXITI (INDUCED.EQUIVALENCE.RELATION
                                                        APPLY-IXIT[ITI] (CANONICAL.PROJECTION RHO S)
                                                        S APPLY-IXITI (MODULO S RHO)) =  RHO

Theorems: * THEOREM TO PROVE : THEOREM 4-12-1 *
        ALL RHO1,RHO2: I
            APPLY-IXITO (EQUIVALENCE.RELATION RHO1 S) AND
            APPLY-IXITO (EQUIVALENCE.RELATION RHO2 S)
            IMPL ((ALL PPHI: ITI SURJECTIVE (PPHI
                                             APPLY-IXITI (MODULO S RHO1)
                                             APPLY-IXITI (MODULO S RHO2)) AND
                  (ALL A : I APPLY-IXITO (IN A S) IMPL
                             APPLY-ITI (PPHI APPLY-ITI
                                              (APPLY-IXIT[ITI] (CANONICAL.PROJECTION RHO1 S) A)) =
                             APPLY-ITI (APPLY-IXIT[ITI] (CANONICAL.PROJECTION RHO2 S) A))) IMPL
                  APPLY-IXITO (SUBSET RHO1 RHO2))

Set of Axiom Clauses Resulting from Normalization
=================================================

          A1:   All x:Any + =(x x)
        * A2:   + APPLY-ITO(set s)
        * A3:   All x:I - APPLY-IXITO(equivalence.relation x s)
                        + MAPPING(apply-ixit[iti](canonical.projection x s) s apply-ixiti(modulo s x))
        * A4:   All x:I - APPLY-IXITO(equivalence.relation x s)
                        + =(apply-[iti]xixiti(induced.equivalence.relation
                                              apply-ixit[iti](canonical.projection x s)
                                              s
                                              apply-ixiti(modulo s x))
                            x)
        * A5:   All x,y:I - APPLY-ITO(set y)  - APPLY-IXITO(equivalence.relation x y)
                          + APPLY-ITO(set apply-ixiti(modulo y x))
        * A6:   All x:Iti y,z:I - APPLY-ITO(set z)  - APPLY-ITO(set y)
                               - SURJECTIVE(x z y)  + MAPPING(x z y)
        * A7:   All x,y,z:Iti u,v:I - APPLY-ITO(set v)  - APPLY-ITO(set u)
                                    - MAPPING(z s v)  - MAPPING(y s u)
                                    - MAPPING(x v u)  + APPLY-IXITO(in f_1(x u y v z) s)
                                    + APPLY-IXITO(
                                        subset
                                        apply-[iti]xixiti(induced.equivalence.relation z s v)
                                        apply-[iti]xixiti(induced.equivalence.relation y s u))
        * A8:   All x,y,z:Iti u,v:I - APPLY-ITO(set v)  - APPLY-ITO(set u)  - MAPPING(z s v)
                                    - MAPPING(y s u)  - MAPPING(x v u)
                                    - =(apply-iti(x apply-iti(z f_1(x u y v z)))
                                        apply-iti(y f_1(x u y v z)))
                                    + APPLY-IXITO(
                                        subset
                                        apply-[iti]xixiti(induced.equivalence.relation z s v)
                                        apply-[iti]xixiti(induced.equivalence.relation y s u))
```

```
Set of Theorem Clauses Resulting from Normalization
===================================================

        * T9: + APPLY-IXITO(equivalence.relation c_1 s)
        * T10: + APPLY-IXITO(equivalence.relation c_2 s)
        * T11: All x:Iti + SURJECTIVE(x apply-ixiti(modulo s c_1) apply-ixiti(modulo s c_2))
        * T12: - APPLY-IXITO(subset c_1 c_2)
        * T13: All x:Iti y:I - APPLY-IXITO(in y s)
                             + =(apply-iti(x apply-iti(apply-ixit[iti](canonical.projection c_1 s) y))
                                 apply-iti(apply-ixit[iti](canonical.projection c_2 s) y))
```

The proof is even more unreadable than ordinary resolution proofs, because of the APPLY-IXITO, APPLY-IXITI, and so on. So we have edited the proof and abbreviated APPLY-IXITO(in x s) by in(x s) and so on. Elsewise the proof is unchanged and originally MKRP-generated. Of course by this method we obtain clauses that are not *first order*, these are the clauses T13 and R1.

```
Edited Refutation in ''Higher Order'' Clauses
=============================================

Initial Clauses:      A1:    All x:Any + =(x x)
                    * A2:    + set(s)
                    * A3:    All x:I - equivalence.relation(x s)
                                     + MAPPING(canonical.projection x s)(s modulo(s x))
                    * A4:    All x:I - equivalence.relation(x s)
                                     + =(induced.equivalence.relation(canonical.projection(x s)
                                                                      s modulo(s x))
                                         x)
                    * A5:    All x,y:I - set(y)  - equivalence.relation(x y)  + set(modulo(y x))
                    * A6:    All x:Iti y,z:I - set(z)  - set(y)  - SURJECTIVE(x z y)  + MAPPING(x z y)
                    * A7:    All x,y,z:Iti u,v:I - set(v)  - set(u)  - MAPPING(z s v)  - MAPPING(y s u)
                                                 - MAPPING(x v u)  + in(f_1(x u y v z) s)
                                                 + subset(induced.equivalence.relation(z s v)
                                                          induced.equivalence.relation(y s u))
                    * A8:    All x,y,z:Iti u,v:I - set(v)  - set(u)  - MAPPING(z s v)  - MAPPING(y s u)
                                                 - MAPPING(x v u)
                                                 - =(x(z(f_1(x u y v z))) y(f_1(x u y v z)))
                                                 + subset(induced.equivalence.relation(z s v)
                                                          induced.equivalence.relation(y s u))
                    * T9:    + equivalence.relation(c_1 s)
                    * T10:   + equivalence.relation(c_2 s)
                    * T11:   All x:Iti + SURJECTIVE(x modulo(s c_1) modulo(s c_2))
                    * T12:   - subset(c_1 c_2)
                    * T13:   All x:Iti y:I - in(y s)  + =(x(canonical.projection(c_1 s)(y))
                                                          canonical.projection(c_2 s)(y))

A7,6 & T13,1  -->   * R1:    All x,y,z,u:Iti v,w:I - set(w)  - set(v)  - MAPPING(u s w)
                                                  - MAPPING(z s v)  - MAPPING(y w v)
                                                  + subset(induced.equivalence.relation(u s w)
                                                           induced.equivalence.relation(z s v))
                                                  + =(x(canonical.projection(c_1 s)(f_1(y v z w u)))
                                                      canonical.projection(c_2 s)(f_1(y v z w u)))

T11,1 & A6,3  -->   * R2:    All x:Iti - set(modulo(s c_1))  - set(modulo(s c_2))
                                       + MAPPING(x modulo(s c_1) modulo(s c_2))

R2,2 & A5,3   -->   * R3:    All x:Iti - set(modulo(s c_1))  + MAPPING(x modulo(s c_1) modulo(s c_2))
                                       - set(s)  - equivalence.relation(c_2 s)

R3,3 & A2,1   -->   * R4:    All x:Iti - set(modulo(s c_1))  + MAPPING(x modulo(s c_1) modulo(s c_2))
                                       - equivalence.relation(c_2 s)

R4,3 & T10,1  -->   * R5:    All x:Iti - set(modulo(s c_1))  + MAPPING(x modulo(s c_1) modulo(s c_2))

R5,1 & A5,3   -->   * R6:    All x:Iti + MAPPING(x modulo(s c_1) modulo(s c_2))  - set(s)
                                       - equivalence.relation(c_1 s)
```

```
R6,2 & A2,1   -->  * R7:    All x:Iti + MAPPING(x modulo(s c_1) modulo(s c_2))
                             - equivalence.relation(c_1 s)

R7,2 & T9,1   -->  * R8:    All x:Iti + MAPPING(x modulo(s c_1) modulo(s c_2))

T10,1 & A5,2  -->  * R9:    - set(s)  + set(modulo(s c_2))

R9,1 & A2,1   -->  * R10:   + set(modulo(s c_2))

T9,1 & A5,2   -->  * R11:   - set(s)  + set(modulo(s c_1))

R11,1 & A2,1  -->  * R12:   + set(modulo(s c_1))

R1,7 & A8,6   -->  * R13:   All x:Iti y,z:I - set(z)  - set(y)
                                            - MAPPING(canonical.projection(c_1 s) s z)
                                            - MAPPING(canonical.projection(c_2 s) s y)
                                            - MAPPING(x z y)
                                            + subset(induced.equivalence.relation(
                                                       canonical.projection(c_1 s) s z)
                                                     induced.equivalence.relation(
                                                       canonical.projection(c_2 s) s y))
                                            - set(z)  - set(y)
                                            - MAPPING(canonical.projection(c_1 s) s z)
                                            - MAPPING(canonical.projection(c_2 s) s y)
                                            - MAPPING(x z y)
                                            + subset(induced.equivalence.relation(
                                                       canonical.projection(c_1 s) s z)
                                                     induced.equivalence.relation(
                                                       canonical.projection(c_2 s) s y))

R13 6=12      -->  * D14:   All x,y:I z:Iti - set(y)  - set(x)
                                            - MAPPING(canonical.projection(c_1 s) s y)
                                            - MAPPING(canonical.projection(c_2 s) s x)
                                            - MAPPING(z y x)
                                            + subset(induced.equivalence.relation(
                                                       canonical.projection(c_1 s) s y)
                                                     induced.equivalence.relation(
                                                       canonical.projection(c_2 s) s x))
                                            - set(y)  - set(x)
                                            - MAPPING(canonical.projection(c_1 s) s y)
                                            - MAPPING(canonical.projection(c_2 s) s x)
                                            - MAPPING(z y x)

D14 5=11      -->  * D15:   All x,y:I z:Iti - set(y)  - set(x)
                                            - MAPPING(canonical.projection(c_1 s) s y)
                                            - MAPPING(canonical.projection(c_2 s) s x)
                                            - MAPPING(z y x)
                                            + subset(induced.equivalence.relation(
                                                       canonical.projection(c_1 s) s y)
                                                     induced.equivalence.relation(
                                                       canonical.projection(c_2 s) s x))
                                            - set(y)  - set(x)
                                            - MAPPING(canonical.projection(c_1 s) s y)
                                            - MAPPING(canonical.projection(c_2 s) s x)

D15 4=10      -->  * D16:   All x,y:I z:Iti - set(y)  - set(x)
                                            - MAPPING(canonical.projection(c_1 s) s y)
                                            - MAPPING(canonical.projection(c_2 s) s x)
                                            - MAPPING(z y x)
                                            + subset(induced.equivalence.relation(
                                                       canonical.projection(c_1 s) s y)
                                                     induced.equivalence.relation(
                                                       canonical.projection(c_2 s) s x))
                                            - set(y)  - set(x)
                                            - MAPPING(canonical.projection(c_1 s) s y)
```

```
D16 3=9       -->  * D17:   All x,y:I z:Iti - set(y)  - set(x)
                                           - MAPPING(canonical.projection(c_1 s) s y)
                                           - MAPPING(canonical.projection(c_2 s) s x)
                                           - MAPPING(z y x)
                                           + subset(induced.equivalence.relation(
                                                      canonical.projection(c_1 s) s y)
                                                    induced.equivalence.relation(
                                                      canonical.projection(c_2 s) s x))
                                           - set(y)  - set(x)

D17 2=8       -->  * D18:   All x,y:I z:Iti - set(y)  - set(x)
                                           - MAPPING(canonical.projection(c_1 s) s y)
                                           - MAPPING(canonical.projection(c_2 s) s x)
                                           - MAPPING(z y x)
                                           + subset(induced.equivalence.relation(
                                                      canonical.projection(c_1 s) s y)
                                                    induced.equivalence.relation(
                                                      canonical.projection(c_2 s) s x))
                                           - set(y)

D18 1=7       -->  * D19:   All x,y:I z:Iti - set(y)  - set(x)
                                           - MAPPING(canonical.projection(c_1 s) s y)
                                           - MAPPING(canonical.projection(c_2 s) s x)
                                           - MAPPING(z y x)
                                           + subset(induced.equivalence.relation(
                                                      canonical.projection(c_1 s) s y)
                                                    induced.equivalence.relation(
                                                      canonical.projection(c_2 s) s x))

T9,1 & A4,1   -->  * R20:   + =(induced.equivalence.relation(canonical.projection(c_1 s)
                                                             s
                                                             modulo(s c_1))
                              c_1)

T10,1 & A4,1  -->  * R22:   + =(induced.equivalence.relation(canonical.projection(c_2 s)
                                                             s
                                                             modulo(s c_2))
                              c_2)

R8,1 & D19,5  -->  * R49:   - set(modulo(s c_1))  - set(modulo(s c_2))
                            - MAPPING(canonical.projection(c_1 s) s modulo(s c_1))
                            - MAPPING(canonical.projection(c_2 s) s modulo(s c_2))
                            + subset(induced.equivalence.relation(
                                       canonical.projection(c_1 s) s modulo(s c_1))
                                     induced.equivalence.relation(
                                       canonical.projection(c_2 s) s modulo(s c_2)))

R49,5 & R22   -->  * RW50:  - set(modulo(s c_1))  - set(modulo(s c_2))
                            - MAPPING(canonical.projection(c_1 s) s modulo(s c_1))
                            - MAPPING(canonical.projection(c_2 s) s modulo(s c_2))
                            + subset(induced.equivalence.relation(
                                       canonical.projection(c_1 s) s modulo(s c_1))
                                     c_2)

RW50,5 & R20  -->  * RW51:  - set(modulo(s c_1))  - set(modulo(s c_2))
                            - MAPPING(canonical.projection(c_1 s) s modulo(s c_1))
                            - MAPPING(canonical.projection(c_2 s) s modulo(s c_2))
                            + subset(c_1 c_2)

RW51,5 & T12,1--> * R52:    - set(modulo(s c_1))  - set(modulo(s c_2))
                            - MAPPING(canonical.projection(c_1 s) s modulo(s c_1))
                            - MAPPING(canonical.projection(c_2 s) s modulo(s c_2))

R52,4 & A3,2  -->  * R53:   - set(modulo(s c_1))  - set(modulo(s c_2))
                            - MAPPING(canonical.projection(c_1 s) s modulo(s c_1))
                            - equivalence.relation(c_2 s)
```

```
R53,4 & T10,1 --> * R54:   - set(modulo(s c_1))  - set(modulo(s c_2))
                           - MAPPING(canonical.projection(c_1 s) s modulo(s c_1))

R54,3 & A3,2  --> * R55:   - set(modulo(s c_1))  - set(modulo(s c_2))  - equivalence.relation(c_1 s)

R55,3 & T9,1  --> * R56:   - set(modulo(s c_1))  - set(modulo(s c_2))

R56,2 & R10,1 --> * R57:   - set(modulo(s c_1))

R57,1 & R12,1 --> * R58:   []


  q.e.d.

Time Used for Refutation: 238 seconds
```

This example shows some shortcomings:

- We have no sorts, so we have to use clauses like + **set(s)**.
- Because we have no partial functions, we cannot express the relation $\Phi \circ \natural_1 = \natural_2$ in the canonical way, but instead we have to write $\forall a_\iota \;\; a \in S \Longrightarrow \Phi(\natural_1(a)) = \natural_2(a)$.
- We cannot write the equivalence relation as a set, that is, as object of type ι, *and* as a binary relation, that is, as objects of type $(\iota \times \iota \rightarrow o)$.

Nevertheless we have here a correct treatment, and an automated theorem prover like the MKRP system can solve the problem in this representation.