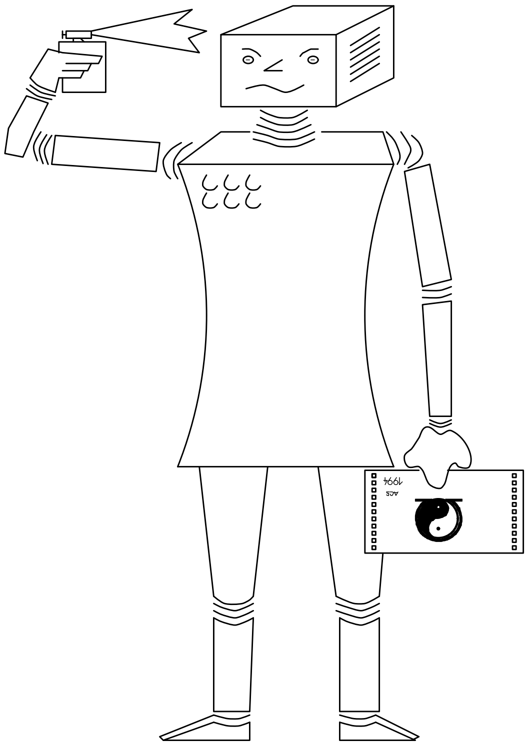

How to Prove Higher Order Theorems in First Order Logic

Manfred Kerber

Published as: In John Mylopoulos and Ray Reiter, eds., *Proceedings of the 12th IJCAI*, Sydney, Australia, p.137–142, Morgan Kaufman, San Mateo, California, USA, 1991.

How to Prove Higher Order Theorems in First Order Logic

Manfred Kerber
Fachbereich Informatik, Universität Kaiserslautern
D-6750 Kaiserslautern, Germany
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 first order logic with flat sorts and 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.

Keywords: higher order logic, second order logic, translation, morphism, soundness, completeness

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

1 Introduction

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 [Andrews *et al.*, 1990] 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 or VON NEUMANN-GÖDEL-BERNAYS have been developed for the formalization of mathematical concepts and for reasoning about them. Other approaches are RUSSEL'S ramified theory of types and CHURCH'S simple theory of types 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 "naïve" set theory than to first order logic. They know about the antinomies and avoid them, for example by the omission 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) define the concepts subset, intersection, union, function, relation, power set, 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:
$\forall + \ \ commutative(+) \iff \forall x, y \ \ x + y \equiv y + x$ or
$\forall P \ \ symmetric(P) \iff (\forall x, y \ \ P(x, y) \Longrightarrow P(y, x))$
Here P is a predicate variable, and *symmetric* is a predicate constant, which expects a predicate term as its argument. This cannot be written immediately in first order logic, because we quantify over P, so P would have to be a variable. On the other hand it must be a predicate because of the expression $P(x, y)$, hence a predicate 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: the induction axiom is second order. Another example of the inadequacy of first order logic comes from the theorem of LÖWENHEIM-SKOLEM: For instance 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.

1.1 Example: A common translation of our formula above in a first order logic is:
$\forall P\ \mathit{symmetric}(P) \iff$
$\quad(\forall x, y\ \mathit{apply}(P, x, y) \implies \mathit{apply}(P, y, x))$
apply is a predicate; it is interpreted freely, although it is intended that it is true exactly when P holds for the other arguments.

The following problems occur:

- 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?

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 Gödel's incompleteness result. In principle such a translation must be equivalent to some set theoretical formulation as stated in Mostowski's isomorphism theorem [Mostowski, 1949].

Related Work

J. van Benthem and K. Doets [1983] 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 introduce 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 [1989], 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 an 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.

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 [Andrews, 1986]. However, we shall write the types in a different way. For example if 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 $\mathcal{L}^\omega$): ι is a type of order 0 that denotes the type of the individuals. o is a type of order 1 and denotes the type of the truth values. 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 types $\tau_1, \ldots, \tau_m$, respectively, and value of type σ. 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 types $\tau_1, \ldots, \tau_m$, respectively.

2.2 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 we often write $\mathcal{L}^\omega$ instead of $\mathcal{L}^\omega(\mathcal{S})$.

2.3 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 types indicated by their subscripts, then we get a term of type σ by $f_{(\tau_1 \times \cdots \times \tau_m \to \sigma)}(t_{\tau_1}, \ldots, t_{\tau_m})$.

2.4 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.

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

The Semantics

The standard semantics is due to Tarski and has been extended by Henkin [1950] 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.

2.6 Definition:

- A *frame* is a collection $\{\mathcal{D}_\tau\}_\tau$ of nonempty sets $\mathcal{D}_\tau$, one for each type τ, such that $\mathcal{D}_o = \{\mathtt{T}, \mathtt{F}\}$ 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*.
- 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 $\mathcal{D}_\tau$.

– An *assignment* into a frame $\{\mathcal{D}_\tau\}_\tau$ is a function ξ that maps each variable of type τ of $\mathcal{L}^\omega$ to an element of $\mathcal{D}_\tau$. 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.

2.7 Definition (Interpretation): An interpretation $\mathcal{M} = \langle\{\mathcal{D}_\tau\}_\tau, \mathcal{J}\rangle$ is a weak interpretation (weak model, general model) for $\mathcal{L}^\omega$ 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
   $\mathcal{V}^\mathcal{M}_\xi(f_{(\tau_1\times\cdots\times\tau_n\to\sigma)}(t_{\tau_1},\ldots,t_{\tau_m})) =$
   $\mathcal{V}^\mathcal{M}_\xi(f_{(\tau_1\times\cdots\times\tau_n\to\sigma)})(\mathcal{V}^\mathcal{M}_\xi t_{\tau_1},\ldots,\mathcal{V}^\mathcal{M}_\xi t_{\tau_m})$
4. $\mathcal{V}^\mathcal{M}_\xi(\varphi\wedge\psi) = \mathcal{V}^\mathcal{M}_\xi\varphi \wedge \mathcal{V}^\mathcal{M}_\xi\psi$ [*]
5. $\mathcal{V}^\mathcal{M}_\xi(\neg\varphi) = \neg\mathcal{V}^\mathcal{M}_\xi\varphi$
6. $\mathcal{V}^\mathcal{M}_\xi(\forall x_\tau\varphi) = \forall d\in\mathcal{D}_\tau \mathcal{V}^\mathcal{M}_{\xi[x_\tau\leftarrow d]}\varphi$

It is a strong interpretation (strong model, standard model), if $\mathcal{D}_\tau = \mathcal{F}(\mathcal{D}_{\tau_1},\ldots,\mathcal{D}_{\tau_m};\mathcal{D}_\sigma)$ for all occurring types τ with $\tau = (\tau_1\times\cdots\times\tau_m\to\sigma)$.

2.8 Remark: It is easy to see that in $\mathcal{L}^1$ for every weak model of a formula set there is a strong model with the same interpretation function $\mathcal{J}$.

Sorted Logics

Now we introduce our target language, a standard many-sorted first order logic with equality predicates on all sorts. Let Σ be a (finite) set of sorts. We define the signature $\mathcal{S}_\Sigma$ 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 $s_1,\ldots,s_m,s\in\Sigma$. In each $\mathcal{S}^{(s,s)}$ we have the binary predicate symbol $\equiv^{(s,s)}$. We index the elements of $\mathcal{S}_\Sigma$ sometimes by their sorts. For instance a function symbol f of sort $(s_1,\ldots,s_m):s$ is written as $f^{(s_1,\ldots,s_m):s}$. Sorted terms and formulae can be defined as usual and we underly the usual semantics. For details see [Kerber, 1990].

The order sorted logic covers this simple situation and therefore the input language of a theorem prover like the Markgraf Karl Refutation Procedure [MKRP, 1984] is well-suited for dealing with the defined logic.

3 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.

3.1 Definition (Morphism of Logics): Let $\mathcal{F}^1$ and $\mathcal{F}^2$ be two logical systems ($\mathcal{L}^\omega$, $\mathcal{L}^n$, 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 $\mathcal{F}^2(\Theta(\mathcal{S}))$.[*]

3.2 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 $\mathcal{F}^2$.

3.3 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 $\mathcal{F}^1$.

3.4 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 formula 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 $\Theta(f(t_1,\ldots,t_m)) = \vartheta(\Theta(f),\Theta(t_1),\ldots,\Theta(t_m))$ with $\vartheta(a,a_1,\ldots,a_m) = \begin{cases} a(a_1,\ldots,a_m) & \text{or} \\ \boldsymbol{\alpha}_a(a,a_1,\ldots,a_m) \end{cases}$
   The constants $\boldsymbol{\alpha}$ 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 $\boldsymbol{\alpha}_a = \Theta(\alpha')$. The case which is chosen can depend only on the a, not on the $a_1,\ldots,a_m$. ($\boldsymbol{\alpha}$ 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 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.

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

Proof: Let $\mathcal{M}$ be a weak model of a formula set Γ, $\mathcal{M} = \langle\{\mathcal{D}_\tau\}_\tau,\mathcal{J}\rangle$ 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

[*]We use the connectives and quantifiers in a naïve way at the meta-level.

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

$\Theta(\varphi)$, where "τ"* denotes the sort upon which the type τ is mapped. We define the sets $\hat{\mathcal{D}}^{"\tau"} := \mathcal{D}_\tau$. $\hat{\mathcal{J}}$ 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. Additionally 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 $\hat{\mathcal{D}}^{"\tau"} = \mathcal{D}_\tau$ 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 $\boldsymbol{\alpha}_a^{"\tau"}$ 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 $f \in \mathcal{D}^{"\tau"}$, for all $x_1 \in \mathcal{D}^{"\tau_1"}, \ldots, x_m \in \mathcal{D}^{"\tau_m"}$ $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}}(\boldsymbol{\alpha}_f^{"\tau"})(f, x_1, \ldots, x_m) := f(x_1, \ldots, x_m)$. Note that $f \in \mathcal{D}^{"\tau"} = \mathcal{D}_\tau$ is a function and hence applicable. Analogously we define the interpretation for the predicates $\boldsymbol{\alpha}_p^{"\tau"}$.

Now $\hat{\mathcal{M}}$ is a model of $\Theta(\varphi)$, which is proved by showing inductively that for all terms and formulae $\mathcal{V}^{\hat{\mathcal{M}}}_{\hat{\xi}} \circ \Theta = \mathcal{V}^{\mathcal{M}}_{\xi}$. The proof is straightforward and can be found in [Kerber, 1990]. ■

4.2 Theorem: If Θ is an injective quasi-homomorphism from $\mathcal{L}^n(\mathcal{S})$ to $\mathcal{L}^1_{sort}(\mathcal{S}_\Sigma)$, 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 $\mathcal{L}^1_{sort}(\mathcal{S}_\Sigma)$. By a sorted version of remark 2.8 there is also a strong model of $\Theta(\Gamma)$. ■

By the theorems above it follows directly that the translation used in example 1.1 is weakly and strongly sound.

4.3 Remark: Note that the formulae that are obtained by these translations are not essentially more difficult than the original ones and that the structure of the formulae (number and position of quantifiers or 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 A Complete Translation

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 restrictions 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 $\hat{\Theta}(\varphi) = \hat{\Theta}'(\varphi) \cup EXT$, where $\hat{\Theta}'$ is a quasi-homomorphism and EXT is the set of extensionality axioms which depends only on the signature. In the following we drop the index n. Again we abbreviate *apply* as $\boldsymbol{\alpha}$.

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

5.1 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 $\mathcal{S}_\Sigma$ 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.

Additionally we have in $\mathcal{S}_\Sigma$ 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 all other formulae we define $\hat{\Theta}'$ as the homomorphic extension.

EXT is the set of the following $\mathcal{L}^1_{sort}$-formulae:

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

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

Now we can define $\hat{\Theta}(\varphi) = \hat{\Theta}'(\varphi) \cup EXT$.

5.2 Remark: It should become obvious now, why we excluded types like $(o \to o)$: For instance let P be a predicate of this type, Q be a predicate of type $(\iota \to o)$,

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

and c be a constant of type ι. Then the translation of $P(Q(c)) \wedge Q(c)$ can be only the apply-construct $\boldsymbol{\alpha}^{"(o\to o)"}(P, \boldsymbol{\alpha}^{"(\iota\to o)"}(Q,c)) \wedge \boldsymbol{\alpha}^{"(\iota\to o)"}(Q,c)$ or the direct translation $P(\boldsymbol{\alpha}^{"(\iota\to o)"}(Q,c)) \wedge \boldsymbol{\alpha}^{"(\iota\to o)"}(Q,c)$. But both formulae are 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 [1983] 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 as functional symbols only the $\boldsymbol{\alpha}^{"\tau"}$; all other symbols are object variables or object constants. Especially the junctor "$\wedge$" has also to be translated to a constant.

5.3 Theorem: $\hat{\Theta}$ is weakly and strongly sound. (For a proof see [Kerber, 1990].)

5.4 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 $\mathcal{D}_\tau$.*
We define $\natural_\tau$ inductively:
$\natural_\iota : \mathcal{D}^{"\iota"} \to \check{\mathcal{D}}_\iota$ is the identity mapping. (This function is obviously bijective.)
Let $\natural_{\tau_i}$ and $\natural_\sigma$ be defined for $\mathcal{D}^{"\tau_1"},\ldots,\mathcal{D}^{"\tau_m"}$ and $\mathcal{D}^{"\sigma"}$. 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:

$$\begin{tikzcd}
\mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"}) : \mathcal{D}^{"\tau"} \arrow[d, "!"', "\natural_\tau"] & \times & \mathcal{D}^{"\tau_1"} & \times\cdots\times & \mathcal{D}^{"\tau_m"} \arrow[r] & \mathcal{D}^{"\sigma"} \arrow[d, "\natural_\sigma"] \\
\check{\mathcal{D}}_\tau \arrow[r, hook] & \mathcal{F}(& \check{\mathcal{D}}_{\tau_1} \arrow[u, "\natural_{\tau_1}^{-1}"'] & ,\ldots, & \check{\mathcal{D}}_{\tau_m} \arrow[u, "\natural_{\tau_m}^{-1}"'] ; & \check{\mathcal{D}}_\sigma)
\end{tikzcd}$$

By the corresponding extensionality axiom in EXT it can be shown that $\natural_\tau$ is bijective (see [Kerber, 1990]).
For $\sigma = o$ and for order of τ is equal to n we define $\natural_\tau(p)(\check{x}_1,\ldots,\check{x}_m) := p(\natural_{\tau_1}^{-1}(\check{x}_1),\ldots,\natural_{\tau_m}^{-1}(\check{x}_m))$, and for order of τ less than n, $\natural_\tau(x)(x_1,\ldots,x_m) := \mathcal{V}_\xi^{\mathcal{M}}(\boldsymbol{\alpha}^{"\tau"})(x, \natural_{\tau_1}^{-1}(\check{x}_1),\ldots,\natural_{\tau_m}^{-1}(\check{x}_m))$ for all $\check{x}_1 \in \check{\mathcal{D}}_{\tau_1}, \ldots, \check{x}_m \in \check{\mathcal{D}}_{\tau_m}$. The bijectivity is shown analogously with help of the extensionality 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.
By induction on the construction of terms it can be proved, that for all terms: $\mathcal{V}_{\check{\xi}}^{\check{\mathcal{M}}} = \natural \circ \mathcal{V}_\xi^{\mathcal{M}} \circ \hat{\Theta}'$ and for all formulae: $\mathcal{V}_{\check{\xi}}^{\check{\mathcal{M}}} = \mathcal{V}_\xi^{\mathcal{M}} \circ \hat{\Theta}'$. The proof can be found in [Kerber, 1990]. 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 $\mathcal{V}_{\check{\xi}}^{\check{\mathcal{M}}}(\varphi) = \mathtt{T}$. ■

5.5 Remark: Because $\hat{\Theta}'$ is an injective quasi-homomorphism, $\hat{\Theta}'^{-1}$ 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 $\mathcal{L}^n$.

5.6 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. Whereas the extensionality axioms are relatively harmless, for *really higher order* theorems it is necessary to add so-called comprehension axioms (compare [Andrews, 1986, p.156]) in order to approximate weak semantics to strong semantics. For many theorems these axioms are not necessary, for the others one must choose the axioms very carefully, otherwise the first order theorem prover will get a search space that is too big. It is the advantage of higher order theorem proving compared to our approach, that there one does not need these axioms (for the prize of the undecidability of unification). In the appendix we give an example of a theorem, where a comprehension axiom is necessary.

6 Summary and Open Problems

In the sections above we introduced the basic machinery for translating higher order formulae to first order logic.

*In order to get strong completeness it would be necessary to achieve bijectivity from $\mathcal{D}^{"\tau"}$ to $\mathcal{F}(\check{\mathcal{D}}_{\tau_1},\ldots,\check{\mathcal{D}}_{\tau_m};\check{\mathcal{D}}_\sigma)$. For $n > 1$ this is impossible because of Gödel's incompleteness theorem.

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 full version of the paper [Kerber, 1990] we generalized the results to logics with equality. An interesting and useful generalization would be to a higher order *sorted* logic. Then the first order logic should 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.

Acknowledgement

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

[Andrews *et al.*, 1990] 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.

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

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

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

[Kerber, 1990] Manfred Kerber. How to prove higher order theorems in first order logic. SEKI Report SR-90-19, Fachbereich Informatik, Universität Kaiserslautern, Kaiserslautern, Germany, 1990.

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

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

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

Appendix

We present an MKRP-proof of CANTOR'S theorem that the power set of a set has greater cardinality than the set itself. We use the formulation of [Andrews, 1986, p.184]. A comprehension axiom is necessary. We write ι as `I`, o as `O`, $\rightarrow$ as `T`, $\boldsymbol{\alpha}^{"(\iota\rightarrow(\iota\rightarrow o))"}$ as `A[IT[ITO]]`, and so on.

```
Formulae given to the editor
Axioms:
* SORT DECLARATIONS *
SORT I,ITO,IT[ITO]:ANY
* TERM DECLARATIONS *
TYPE A[ITO](ITO I)
TYPE A[IT[ITO]](IT[ITO] I):ITO
TYPE SUBSET(ITO ITO)
* DEFINITION SUBSET *
ALL A,B:ITO SUBSET(A B) EQV
          (ALL X:I A[ITO](A X) IMPL A[ITO](B X))
* COMPREHENSION AXIOM *
ALL S:ITO ALL G:IT[ITO] EX P:ITO
(ALL X:I A[ITO](P X) EQV
 (A[ITO](S X) AND (NOT A[ITO](A[IT[ITO]](G X) X))))

Theorems:
ALL S:ITO (NOT EX G:IT[ITO] ALL F:ITO
SUBSET (F S) IMPL (EX J:I A[ITO](S J) AND
                                 A[IT[ITO]](G J) = F))

Refutation:
A1: All x:Any + =(x x)
A2: All x:I y:It[ito] z:Ito - A[ITO](f_1(z y) x)
                            + A[ITO](z x)
A3: All x:I y:It[ito] z:Ito - A[ITO](f_1(z y) x)
                            - A[ITO](a[it[ito]](y x) x)
A4: All x:I y:It[ito] z:Ito + A[ITO](f_1(z y) x)
           - A[ITO](z x)   + A[ITO](a[it[ito]](y x) x)
T5: All x:Ito + A[ITO](x f_2(x))  + A[ITO](c_1 f_3(x))
T6: All x:Ito + A[ITO](x f_2(x))
              + =(a[it[ito]](c_2 f_3(x)) x)
T7: All x:Ito - A[ITO](c_1 f_2(x))  + A[ITO](c_1 f_3(x))
T8: All x:Ito - A[ITO](c_1 f_2(x))
              + =(a[it[ito]](c_2 f_3(x)) x)

T5,1 & A2,1   -> R8:  All x:It[ito] y:Ito
                      + A[ITO](c_1 f_3(f_1(y x)))
                      + A[ITO](y f_2(f_1(y x)))
R8,2 & T7,1   -> R9:  All x:It[ito]
                      + A[ITO](c_1 f_3(f_1(c_1 x)))
                      + A[ITO](c_1 f_3(f_1(c_1 x)))
R9 1=2        -> D10: All x:It[ito]
                      + A[ITO](c_1 f_3(f_1(c_1 x)))
T6,1 & A2,1   -> R12: All x:It[ito] y:Ito
                      + =(a[it[ito]](c_2 f_3(f_1(y x)))
                         f_1(y x))
                      + A[ITO](y f_2(f_1(y x)))
R12,2 & T8,1 -> R13: All x:It[ito]
                      + =(a[it[ito]](c_2 f_3(f_1(c_1 x)))
                         f_1(c_1 x))
                      + =(a[it[ito]](c_2 f_3(f_1(c_1 x)))
                         f_1(c_1 x))
R13 1=2       -> D14: All x:It[ito]
                      + =(a[it[ito]](c_2 f_3(f_1(c_1 x)))
                         f_1(c_1 x))
D14,1 & A3,2 -> P15: All x:Ito y:It[ito]
                      - A[ITO](f_1(c_1 y) f_3(f_1(c_1 y)))
                      - A[ITO](f_1(x c_2) f_3(f_1(c_1 y)))
P15 (factor) -> F16: - A[ITO](f_1(c_1 c_2)
                             f_3(f_1(c_1 c_2)))
A4,1 & F16,1 -> R17: - A[ITO](c_1 f_3(f_1(c_1 c_2)))
                      + A[ITO](a[it[ito]](c_2
                                   f_3(f_1(c_1 c_2)))
                             f_3(f_1(c_1 c_2)))
R17,2 & D14  -> RW18:- A[ITO](c_1 f_3(f_1(c_1 c_2)))
                      + A[ITO](f_1(c_1 c_2)
                             f_3(f_1(c_1 c_2)))
RW18,2 & P15,2-> R19:- A[ITO](c_1 f_3(f_1(c_1 c_2)))
                      - A[ITO](f_1(c_1 c_2)
                             f_3(f_1(c_1 c_2)))
R19,2 & RW18,2-> R20:- A[ITO](c_1 f_3(f_1(c_1 c_2)))
                      - A[ITO](c_1 f_3(f_1(c_1 c_2)))
R20 1=2       -> D21: - A[ITO](c_1 f_3(f_1(c_1 c_2)))
D21,1 & D10,1-> R22: []
```