Interpretation of a Mizar-like Logic in First Order Logic

Ingo Dahn*
University of Koblenz-Landau
Department of Computer Science

1 Introduction

Automated theorem provers for first order logic have reached a state where they can give useful support for interactive theorem proving. However, most real world problems handled in interactive theorem proving are formulated in a typed language. First order provers have currently rather limited capabilities to handle types. Therefore type information has to be encoded in an efficient way. What is most efficient, depends from the type system as well as from the first order prover at hand.

In this paper we describe a general purpose interpretation of a large fragment of the typed logic used in the Mizar Mathematical Library ([Rud92], [Try93]into untyped first order logic. This poses also new challenge problems for first order automated provers (see [DahWer97]). A general definition of an interpretation based on concepts from abstract model theory sets the theoretical framework.

2 Semantic Foundations

Libraries of theorems are basically collections of sentences that are assumed to be true in a given class of models. In this abstract setting, automated theorem provers provide a potential library - the library of all formulas they can prove. The correctness proof for the calculus underlying a specific prover provides evidence that all formulas in this potential library are true in the class of all their models.

Actual libraries are dynamic. Over time, new formulas are proved and added to the library, other formulas may be removed because they are subsumed by new theorems. Also the language is changed by introducing new concepts.

However, the semantics of the formulas is fixed. Hence it can serve as a basis for the consistent combination combination of knowledge from various sources. Abstract model theory has provided a theoretical framework to study semantic interrelations between several deductive systems. Therefore, we give slight generalizations of its most basic definitions from [Ba74].

Abstract model theory has abstracted from the syntax of a particular language. The only essential property of a logic in this setting is, to determine whether a particular formula is valid in a particular model. At this stage we do not care about how the logic determines validity in detail. We only want to be a little more specific on formulas and

*Rheinau 1, D-56075 Koblenz, dahn@uni-koblenz.de, supported in part by the Deutsche Forschungsgemeinschaft within the Schwerpunkt "Deduktion"

models. These are connected by the concept of a signature. Again, there is currently no need to define what a signature is. We simply note that each *logic* $\mathcal{L}$ accepts a specific set $\Sigma_{\mathcal{L}}$ of signatures. For example, some logics may require all signatures to include special symbols like $=$ or $\in$. For each signature $\sigma \in \Sigma_{\mathcal{L}}$ the logic fixes a class of models $Mod_{\sigma}^{\mathcal{L}}$, a set of formulas $\Phi_{\sigma}^{\mathcal{L}}$ and a relation $\models_{\sigma}^{\mathcal{L}}$which determines, whether a formula $H \in \Phi_{\sigma}{}^{\mathcal{L}}$ holds in a model $M \in Mod_{\sigma}^{\mathcal{L}}$ ($M \models_{\sigma}^{\mathcal{L}} H$).

Then, an *$\mathcal{L}$-theory* of signature σ is simply a subset of $\Phi_{\sigma}^{\mathcal{L}}$. This is sufficient to define the concepts of models and semantic consequence for each such abstract logic $\mathcal{L}$.

Definition 1 *For each $\mathcal{L}$-theory T of signature σ the class of all* $\mathcal{L}$-models *of T is the class $Mod_{\sigma}^{\mathcal{L}}(T)$ of all M in $Mod_{\sigma}^{\mathcal{L}}$, such that $M \models_{\sigma}^{\mathcal{L}} H$ for all $H \in T$.*

This induces the consequence relation, denoted also by $\models_{\sigma}^{\mathcal{L}}$:

Definition 2 *A formula $A \in \Phi_{\sigma}^{\mathcal{L}}$ is a* consequence *of a theory $T \subseteq \Phi_{\sigma}^{\mathcal{L}}$ ($T \models_{\sigma}^{\mathcal{L}} A$) if and only if $M \models_{\sigma}^{\mathcal{L}} A$ for each model M from $Mod_{\sigma}^{\mathcal{L}}(T)$.*

We are well aware that so far the consequence relation depends on the logic as well as on the signature. Therefore, abstract model theory introduces additional conditions on the persistence of the validity relation under signature extensions. However, for our purposes, we shall not need these conditions.

3 Interpretations

In order to use theorems proved by one system (the source system) to enhance the knowledge of another system (the target system), the logic of the second system must be interpreted in the logic of the first system. From the point of view of system architecture this is a mediation service [WiGe97].

For our intended application we think of the source system as an automated prover with logic $\mathcal{S}$, while the target system is a library of formulas from a logic $\mathcal{T}$. However, the same considerations can be applied in order to combine the libraries of two interactive theorem provers as well.

We have to interpret proof problems from $\mathcal{T}$ as proof problems in $\mathcal{S}$ that can be solved by the source system. Our interpretation has to ensure, that the consequences proved by the source system pull back to valid consequences in the target logic. In order to be useful, it is not necessary that the source is able to handle *all* knowledge that the target system can handle - some interesting subset suffices and there can be required some translation procedure ν between the formulas of the source logic and that of the target logic. Also, the models of the two logics can be quite different (for example think of an interpretation of a geometric model like a plane as an arithmetic model consisting of pairs of Cartesian coordinates). This interpretation μ of the models requires some translation ι of between their signatures.

The following definition provides the concept of interpretation with an exact meaning.

Definition 3 *An* interpretation *I of a logic $\mathcal{T}$ in a logic $\mathcal{S}$ consists of a set $\Sigma^I \subseteq \Sigma_{\mathcal{T}}$ of signatures, for each $\sigma \in \Sigma^I$ an $\mathcal{S}$-theory $\Theta_{I,\sigma}$ and three mappings:*

- ι maps a set of signatures Σ^I into $\Sigma_{\mathcal{S}}$,

- μ maps $Mod^{\mathcal{T}}_{\sigma}$ into $Mod^{\mathcal{S}}_{\iota(\sigma)}(\Theta_{I,\sigma})$ for each $\sigma \in \Sigma^I$,
- ν maps for each $\sigma \in \Sigma^I$ a subset of $\Phi^{\mathcal{T}}_{\sigma}$ into $\Phi^{\mathcal{S}}_{\iota(\sigma)}$

such that for all $\sigma \in \Sigma^I$, for each sentence H in the domain of ν and for each model M from $Mod^{\mathcal{T}}_{\sigma}$

$$M \models^{\mathcal{T}}_{\sigma} H \text{ if and only if } \mu(M) \models^{\mathcal{S}}_{\iota(\sigma)} \nu(H).$$

The second of these conditions requires, that the models in the range of μ satisfy some set of conditions $\Theta_{I,\sigma}$ which can be stated in the source logic.

Example 1 *Let the source logic $\mathcal{S}$ be equational logic and let the target logic $\mathcal{T}$ be full first order logic. We obtain an interpretation I by taking Σ_I as the set of signatures containing equality, ι as the operation, that deletes all predicates except equality, μ as the giving the reduct of a model to its equational part, $\Theta_{I,\sigma}$ as the empty theory and ν as the identity on equational first order sentences.*

Example 2 *Take 2-sorted first order logic as source logic $\mathcal{S}$, full monadic second order logic as target logic $\mathcal{T}$ and $\Sigma^I = \Sigma_{\mathcal{T}}$ as the set of all signatures. ι extends each first order signature by a second sort set, μ extends each first order model by adding the powerset of its universe as a second sort and the set theoretic $\in$ as a new relation between elements of the universe and elements of sort set, i. e. subsets of the universe. These 2-sorted models will satisfy a number of 2-sorted first order conditions that can be put into the theory $\Theta_{I,\sigma}$, for example the axiom of extensionality or the collection schema*

$$\exists X : set\ \forall y\,(y \in X \leftrightarrow H(y)).$$

In this example, not all $\mathcal{S}$-models of $\Theta_{I,\sigma}$ will be in the range of μ, for example by the Löwenheim-Skolem theorem there are models of this theory where there are countably many objects of sort set, hence the sort of sets cannot be a full powerset in these models. Models of $\Theta_{I,\sigma}$ are called weak models *in the theoretical foundations of higher order theorem provers.*

Example 3 *Let $\mathcal{S}$ be a first order logic with a designated binary symbol $\prec$. $\mathcal{T}$ is now ordinary first order logic, not using $\prec$ in any of its signatures. For M a $\mathcal{T}$-model of signature σ, let $\mu(M)$ be an expansion of M to $\sigma \cup \{\prec\}$ by interpreting $\prec$ as a well-ordering of the universe. It is a well-known consequence of the axiom of choice that this is always possible. Then, the schema of transfinite induction*

$$\forall x\,(\forall y\,(y \prec x \rightarrow H(y)) \rightarrow H(x)) \rightarrow \forall z\,(H(z))$$

can be included in the theory $\Theta_{I,\sigma}$ for each formula H of signature $\sigma \cup \{\prec\}$ and can be used by theorem provers for the source logic $\mathcal{S}$. Again, there will be weak models, i. e. models of $\Theta_{I,\sigma}$, where the ordering $\prec$ is not a well-founded.

The following theorem states that the existence of an interpretation of a target logic $\mathcal{T}$ in source logic $\mathcal{S}$ justifies the use of translations of theorems from $\mathcal{S}$ in $\mathcal{T}$.

Theorem 1 *Let I be an interpretation of the logic $\mathcal{T}$ in the logic $\mathcal{S}$. Let $\sigma \in \Sigma^I$, $\Gamma \subseteq \Phi^{\mathcal{T}}_{\sigma}$, $A \in \Phi^{\mathcal{T}}_{\sigma}$. Then $\nu(\Gamma) \cup \Theta_{I,\sigma} \models^{\mathcal{S}}_{\iota(\sigma)} \nu(A)$ implies that $\Gamma \models^{\mathcal{T}}_{\sigma} A$, where $\nu(\Gamma)$ denotes the image of Γ under ν.*

Proof. If M is a $\mathcal{T}$-model of Γ, then $\mu(M)$ must be an $\mathcal{S}$-model of $\nu(\Gamma) \cup \Theta_{I,\sigma}$. Hence, $\mu(M)$ is also a model of $\nu(A)$ and therefore M must be a model of $\nu(A')$.**qed**

We mention that the converse of the theorem holds under the additional assumption that there are no weak models, i.e. $Mod^{\mathcal{S}}_{\iota(\sigma)}(\Theta_{I,\sigma})$ is the image of $Mod^{\mathcal{T}}_{\sigma}$ under μ.

Having an interpretation of a logic $\mathcal{T}$ in a logic $\mathcal{S}$ does not mean that there is a procedure to translate proofs from a calculus for $\mathcal{S}$ into proofs in a calculus for $\mathcal{T}$. Thus, in the third example above, proofs in a calculus for $\mathcal{S}$ can make use of transfinite induction, which cannot be translated directly in ordinary first order logic. If a calculus for $\mathcal{T}$ is complete, then it can merely be said that there must be a proof of the sentence A from Γ in this calculus.

4 Basic Properties of the Mizar Mathematical Library

The MIZAR MATHEMATICAL LIBRARY is a collection of mathematical papers (*articles*), written in the MIZAR language. This library has evolved over more than 10 years and consists of more than 20.000 theorems.

All theorems in the MIZAR MATHEMATICAL LIBRARY are proved from the axioms of TARSKI-GROTHENDIECK set theory. This is a set theoretic system, stronger than the more familiar system of ZERMELO and FRAENKEL with the axiom of choice. Especially, there is an unbounded class of strongly inaccessible cardinals. TARSKI-GROTHENDIECK set theory is formulated in first order logic using variables for objects of a single type - *set*. Basic symbols are only the equality symbol $=$ and the membership symbol $\in$. MIZAR treats also the real numbers, the natural numbers and the arithmetic operations as primitive. However, from a theoretical point of view, these could be introduced as derived concepts.

There are some tools to introduce new types - called modes - in MIZAR. All these types are subtypes of *set*, i.e. ultimately, every object occurring in formulas in the MIZAR MATHEMATICAL LIBRARY is a set. Since $=$ and $\in$ take arbitrary arguments of type *set*, there can occur also arguments of all other types on both sides of these symbols.

Given types can be restricted by additional properties, called attributes. When a type S_1 is introduced by restricting a type S (the mother mode of S_1 in MIZAR terminology) by attributes A_1,...,A_n, this means that S_1 is exactly the type of all objects of type S that satisfy the additional conditions A_1,...,A_n. This can also be paraphrased by saying that A_1,...,A_n are the conditions that permit to consider an object of type S as having also the subtype S_1. The meaning of the attributes must have been defined before. The characteristic property

$$\forall (X : S)(A_1(X) \wedge \ldots \wedge A_n(X) \leftrightarrow \exists (Y : S)(Y = X)) \tag{1}$$

can be used only when it has been proved that there is an object of type S that satisfies A_1,...,A_n.

Whenever a denotes a set, it is possible to introduce the type $element\_of(a)$. E.g. the type of real numbers is constructed in this way from the set of real numbers. Whenever $element\_of(a)$ is introduced by the user, MIZAR generates the obligation to prove that a is nonempty. The characteristic property

$$\forall (X : set)(X \in a \leftrightarrow \exists (Y : element\_of(a))(Y = X)) \tag{2}$$

can be used only when this proof obligation has been satisfied.

a used in the example above, can be a term with parameters. E.g. a can be $\wp(X)$ where $\wp$ denotes the power set functor and X is a variable for sets. Then it is possible to prove a sentence like

$$\forall X\ (\forall (Y : element\_of\ (\wp(X)))\ Y \subseteq X).$$

In this way, type constructors can have object parameters. All variables must be bound by quantifiers. This applies especially to variables which occur as parameters in terms that denote types (modes). For the following, it is important to note that terms denoting types can have only object variables - there are no type variables in MIZAR. Nevertheless, type constructors known from type systems of other logics, can be modelled even in a restricted subsystem of MIZAR.

From a set theoretic point of view, types in MIZAR denote classes. Some of these classes are so small that they can be represented as sets. We may call a type S small if

$$\exists (X : set) \forall (Y : S)\ (Y \in X)$$

can be proved. Then, by the collection schema, it can be proved that there is some a such that objects of type S are exactly the objects of type $element\_of\ (a)$.

Thus when S_1, S_2 are small types represented as

$$element\_of\ (a_1), element\_of\ (a_2),$$

it can be proved that the class of all functions taking arguments of type S_1 and having values of type S_2 is small, i.e. there is a set $function\ (a_1, a_2)$ which describes exactly the set all function from a_1 into a_2. Hence this type can be introduced as

$$element\_of\ (function\ (a_1, a_2)).$$

5 Interpretations of Mizar Formulas in First Order Logic

Currently, the most advanced automated theorem provers take formulas in untyped first order logic as input. In order to apply them to extend a library of formulas, the logic of the library must be interpreted in untyped first order logic. We propose some way to do this for the MIZAR MATHEMATICAL LIBRARY.

There is a naive interpretation of the MIZAR logic in first order logic by expanding all definitions. For example, $\forall X : element\_of\ (a)\ H(X)$ translates into

$$\forall X : set\ (X \in a \rightarrow H(X)).$$

Given a type S_1 as the restriction of the type S by the attributes $A_1, \ldots, A_n$ as in the last section, then a Mizar formula $\forall X : S_1\, H(X)$ could be translated into

$$\forall Y : S\ (A_1(Y) \wedge \ldots \wedge A_n(Y) \rightarrow H(Y)).$$

This process could be continued until there remain only variables of type set. When the resulting formulas are handed over to a first order theorem prover, the prover has to solve many proof obligations to ensure the type correctness conditions. For example, to prove $H(a)$ for an object a of type S_1 from the assumption $\forall X : S_1\, H(X)$, the assumptions $A_1(a), \ldots, A_n(a)$ have to be confirmed. This creates heavy deductive overload, especially for provers working with depth bound strategies.

However, type checking can be done algorithmically in the MIZAR type system. More precisely, each term t must be declared in Mizar with a unique minimal type τ. Of course, t will also have all types of which τ is a restriction. It is also possible, that t can be proved to be equal to an object of a different type. For example, it can be proved that empty lists of elements of different types are equal. This way of reconsidering an object as an object of a different type must be justified by a proof also in MIZAR.

The following interpretation intends to carry over algorithmic typechecking from MIZAR to automated theorem provers by encoding type information into terms in order to prevent the generation of additional proof problems due to type checking obligations. The interpretation will work on a large class of formulas from the MIZAR logic (not only on clauses). Running a resolution prover on an interpreted MIZAR theory will yield a unification failure when the prover tries to bind a term to a variable which does not have an appropriate type. Restrictions of this method are discussed in the last Section. In the following, the MIZAR logic takes the role of the target logic $\mathcal{T}$ and first order logic is the source logic $\mathcal{S}$.

Models of the logic of the MIZAR MATHEMATICAL LIBRARY are models of GROTHENDIECK-TARSKI set theory. Subsequently we assume that such models exist, i. e. that the MIZAR logic is consistent. These models are augmented by types for objects of MIZAR modes. From a semantic point of view, these types are predicates with a designated argument for the objects of the given type and potentially other arguments as parameters.

set is the top type of the MIZAR type system. Consequently, the universe of models of the MIZAR logic consists of all objects of type *set*. Functions are special sets. Hence the semantics of the type of all functions is given by a predicate which selects all objects of type *set* that satisfy a certain property (being $function-like$). If a and b are objects of type *set*, it is possible to define the type $function(a,b)$ of all functions from a into b by a predicate that selects those functions with domain in a and range being a subset of b.

MIZAR requires the user to prove for each type constructor that for each instance of the parameter arguments with objects of appropriate types in any model of GROTHENDIECK-TARSKI set theory, there is some object satisfying this predicate in this model, i. e. types defined by MIZAR modes will be always non-empty.

Let M be a model of TARKI-GROTHENDIECK set theory and let

$$S\left(x,u_1,\ldots,u_n\right)$$

be a MIZAR mode, seen as a predicate with argument x and parameters $u_1,\ldots,u_n$. For all $a_1,\ldots,a_n \in |M|$, where $|M|$ denotes the universe of M.

$$\{x \in |M| : M \models_\sigma^{\mathcal{T}} S\left(x,a_1,\ldots,a_n\right)\} \neq \emptyset.$$

Hence, there is an $n+1$-ary function f_S on $|M|$ such that

$$M \models_\sigma^{\mathcal{T}} S\left(f_S\left(a,a_1,\ldots a_n\right),a_1,\ldots,a_n\right)$$

for all $a,a_1,\ldots,a_n \in |M|$ and $f_S\left(a,a_1,\ldots,a_n\right)=a$ if $M \models S\left(a,a_1,\ldots,a_n\right)$. This means that $f_S\left(x,a_1,\ldots,a_n\right)$ maps $|M|$ into the interpretation of $S\left(x,a_1,\ldots,a_n\right)$ in M and is the identity on the set of elements that satisfy this relation. The following Lemma is an immediate consequence of this definition of f_S.

Lemma 1 *Let* $U=\{x \in |M| : M \models_\sigma^{\mathcal{T}} S\left(x,a_1,\ldots,a_n\right)\}$ *and let* f_S *also denote the unary function defined by* $f_S\left(x,a_1,\ldots,a_n\right)$. *Then*

- U is the range of f_S,
- f_S is idempotent on U,
- the following condirions are equivalent for all $x \in |M|$:
  1. $x \in U$,
  2. $f_S(x) = x$.
  3. x is an object of type S with parameters $a_1, \ldots, a_n$.

A MIZAR signature σ consists - beside the mode of all sets and the relation symbols $=$ and $\in$ - of a finite set of user defined MIZAR modes, relations and functors. Our interpretation does not need a restriction on the admissible signatures, i. e. $\Sigma^I = \Sigma_{\mathcal{T}}$. For each MIZAR signature σ let $\iota(\sigma)$ be the first order signature which contains beside $=$ and $\in$ these symbols and new $n+1$-ary functors f_S for each MIZAR mode S from σ with n parameters.

If M is a model of such a signature σ, then let $\mu(M)$ be the first order model of signature $\iota(\sigma)$ with the same universe as M, $\in$ as in M, and the remaining relations and functors defined as follows.

Relations are defined as in M for arguments in their specific domains. If one of the arguments in $\mu(M)$ is outside this domain, we fix a truth value for this relation in an arbitrary way.

Similarly, functors are interpreted in $\mu(M)$ on their domains in M as in M and in an arbitrary but fixed way for the remaining arguments.

The new functors f_S are interpreted as described above. This completes the description of the mappings ι and μ. We complete our definition of an interpretation of the MIZAR logic into first order logic by giving the description of the mapping ν which translates MIZAR formulas into first order formulas and the description of $\Theta_{I,\sigma}$.

In fact, we shall extend ν to work on arbitrary terms in the MIZAR logic. When X is a variable of a type S with parameters $u_1, \ldots, u_n$ let

$$\nu(X) = f_S(X, \nu(u_1), \ldots, \nu(u_n)).$$

Whenever f is a functor of σ taking arguments of type $S_1, \ldots, S_n$ and declared as giving values of type S, $t_1, \ldots, t_n$ are terms of type $S_1, \ldots, S_n$ respectively, we define

$$\nu(f(t_1, \ldots, t_n)) = f_S(f(\nu(t_1), \ldots, v(t_n))).$$

Especially, if f is a constant, then $\nu(f) = f_S(f)$. Note that by the definition of f_S above

$$\mu(M) \models^{\mathcal{S}}_{\sigma} f_S(f) = f.$$

If r is a predicate, then let

$$\nu(r(t_1, \ldots, t_n)) = r(\nu(t_1), \ldots, \nu(t_n)).$$

i.e. relation symbols are not changed. This is of special importance for the equality predicate, since it gives provers a chance to utilize their special treatments of equality. ν distributes over quantifiers and propositional operators. Especially, variables following

a quantifier remain unchanged. The theory $\Theta_{I,\sigma}$ contains beside the axioms of TARSKI-GROTHENDIECK set theory additional informations on the declarations in use. When f is a functor as above, we add to $\Theta_{I,\sigma}$ the universal closure of

$$f_S\left(f\left(v\left(X_1\right),\ldots,\nu\left(X_n\right)\right)\right) = f\left(\nu\left(X_1\right),\ldots,\nu\left(X_n\right)\right) \tag{3}$$

where $X_1,\ldots,X_n$ are variables of type $S_1,\ldots,S_n$ respectively. Moreover we add

$$\forall X\, f_{set}\left(X\right) = X$$

and

$$\forall X\, f_S\left(f_S\left(X\right)\right) = f_S\left(X\right)$$

When the type S_1 is defined by restricting the type S by the attributes A_1,...,A_n, we add

$$\begin{aligned} &\forall X\ \left(A_1\left(f_S\left(X\right)\right) \wedge \ldots \wedge A_n\left(f_S\left(X\right)\right) \leftrightarrow f_{S_1}\left(f_S\left(X\right)\right) = f_S\left(X\right)\right) \\ &\forall X\ \left(f_S\left(f_{S_1}\left(X\right)\right) = f_{S_1}\left(X\right)\right) \end{aligned} \tag{4}$$

When S is defined as $element\_of\left(a\right)$,

$$\forall X\left(X \in \nu\left(a\right) \leftrightarrow f_S\left(X\right) = X\right)$$

is added to $T'_{I,\sigma}$.

Theorem 2 *I as defined above by the mappings ι, μ, ν and the first order theories $\Theta_{I,\sigma}$ is an interpretation of the* MIZAR *logic into untyped first order logic.*

Proof. The first thing to show is, that first order models in the range of μ satisfy $\Theta_{I,\sigma}$. Since $\in$ and $=$ are not changed by μ, a model $\mu\left(M\right)$ must be a model of TARSKI-GROTHENDIECK set theory, since the MIZAR model M is.

If f is a functor symbol that is declared to take arguments of type $S_1,\ldots,S_n$ and to yield a value of type S, 3 expands into

$$f_S\left(f\left(f_{S_1}\left(X_1\right),\ldots,f_{S_n}\left(X_n\right)\right)\right) = f\left(f_{S_1}\left(X_1\right),\ldots,f_{S_n}\left(X_n\right)\right)$$

(we discard parameters of the argument types to simplify the notation). Now let $a_1,\ldots,a_n$ be arbitrary elements of $\left|\mu\left(M\right)\right| = \left|M\right|$. Then for $i = 1\ldots n$ we have $S_i\left(f_{S_i}\left(a_i\right)\right)$ by the definition of f_{S_i}. Hence $f_{S_1}\left(a_1\right),\ldots,f_{S_n}\left(a_n\right)$ are in the domain of f and hence $S\left(f\left(f_{S_1}\left(a_1\right),\ldots,f_{S_n}\left(a_n\right)\right)\right)$. Therefore, since f_S is defined to be the identity on S,

$$f_S\left(f\left(f_{S_1}\left(a_1\right),\ldots,f_{S_n}\left(a_n\right)\right)\right) = f\left(f_{S_1}\left(a_1\right),\ldots,f_{S_n}\left(a_n\right)\right).$$

Since the hole universe of M consists of elements of type *set* and this must be the range of f_{set}. The functions f_S are the identity on their respective ranges, hence f_{set} is the identity.

Now suppose S_1 is defined by restricting the type S by the attributes A_1,...,A_n. Let $a \in \left|M\right|$ be arbitrary and let $b = f_S\left(a\right)$. Hence

$$M \models^{\mathcal{T}}_{\sigma} \left(A_1\left(b\right) \wedge \ldots \wedge A_n\left(b\right) \leftrightarrow \exists X : S_1\, b = x\right)$$

by (1). The right hand side of this equivalence means that $S_1\left(b\right)$ holds and is by Lemma 1 also equivalent with $f_{S_1}\left(b\right) = b$. Now, replacing b by $f_S\left(A\right)$, we confirm (4). b is also

a member of the set of all objects of type S, on which f_S is idempotent. Hence, also $f_S(f_{S_1}(a)) = f_{S_1}(a)$. The last sentence of $\Theta_{I,\sigma}$ reflects similarly (2).

The remaining third condition of the definition of an interpretation states, that the translation ν does not change the truth value of MIZAR sentences. First we show by induction on the form of t that for each legal MIZAR term t and for each instantiation of its variables the value of t in M equals the value of $\nu(t)$ in $\mu(M)$. More precisely, a nested induction is required, where the outer induction goes over the number of parameters occurring in the involved types.

If t is a variable of sort S, then the value of t satisfies the predicate characterizing S, hence $t = f_S(t) = \nu(t)$. If t is a compound term $f(t_1, \ldots, t_n)$ where f is declared as above, then by induction hypothesis

$$\nu(t) = f_S(f(\nu(t_1), \ldots, v(t_n))) = f_S(f(t_1, \ldots, t_n)) = f(t_1, \ldots, t_n) = t$$

by Lemma 1.

The truth value of atomic formulas has not been changed for arguments of the sorts admitted according to the MIZAR declarations. Only such arguments occur in translations of legal MIZAR formulas by ν. The induction over propositional connectives is trivial.

Now, consider a MIZAR formula $\exists X : S\, H(X)$.

$$\nu(\exists X : S\, H(X)) = \exists X\, \nu(H(X)).$$

If $M \models^{\mathcal{T}}_{\sigma} \exists X : S\, H(X)$, say $M \models^{\mathcal{T}}_{\sigma} H[a/X]$ for some a such that $S(a)$, then $\mu(M) \models^{\mathcal{S}}_{\iota(\sigma)} \nu(H)[a/X]$, Hence $\mu(M) \models^{\mathcal{S}}_{\iota(\sigma)} \exists X\, \nu(H(X))$. Conversely, assume that $\mu(M) \models^{\mathcal{S}}_{\iota(\sigma)} \exists X\, \nu(H(X))$, say $\mu(M) \models^{\mathcal{S}}_{\iota(\sigma)} \nu(H(a))$. a need not be of sort S, but we can observe, that X occurs inside H always inside f_S. Moreover by Lemma 1 $f_S(f_S(a)) = f_S(a)$ Therefore, the element $b = f_S(a) = f_S(b)$ is of sort S and we have that also $\mu(M) \models^{\mathcal{S}}_{\iota(\sigma)} \nu(H(b))$, hence $M \models^{\mathcal{T}}_{\sigma} H(b)$. This yields $M \models^{\mathcal{T}}_{\sigma} \exists X : S\, H(X)$. The universal quantifier can be treated similarly.**qed**

6 Modifications

Consider the following sentence which states that each relation can be extended.

$$\forall A : set \forall B : set \forall C : set(A \subseteq B \rightarrow \forall F : relation(A, C) \exists G : relation(B, C) F \subseteq G).$$

Let $f_s.f_r$ denote the function symbols introduced for the type constructors *set* and *relation*. Then this sentence is translated by the method described in the last section into

$$\forall A \forall B \forall C(f_s(A) \subseteq f_s(B) \rightarrow \forall F \exists G f_r(F, f_s(A), f_s(C)) \subseteq f_r(G, f_s(B), f_s(C))).$$

Note that this translation is performed prior to generating clauses for a prover. Since f_s is the identity function, it can be omitted. The resulting formula yields the clause

$$A \subseteq B \rightarrow f_r(F, A, C) \subseteq f_r(s(A, B, C, F), B, C),$$

where s is a new Skolem function.

Suppose we want to infer that each functions can be extended in a similar way to a relation, where functions are defined as special relations. f_f denotes the function symbol introduced for the function type constructor. The negated goal gives the clauses

$$a \subseteq b$$

$$\neg f_f(h(a,b,c),a,b) \subseteq f_r(G,b,c)$$

for new Skolem symbols a, b, c, h. Since relations are functions, the theory describing the interpretation yields

$$f_r(f_f(X,A,B),A,B) = f_f(X,A,B).$$

Hence, f_f can be replaced already in the non-clause form by a term starting with f_r. Then a simple resolution step completes the proof.

On the other hand, as expected, it cannot be proved from the above formula that each function can be extended similarly to a function with an extended domain.

7 Discussion

The interpretation we have given encodes type information into first order terms. It is a general purpose interpretation working for any target first order prover. It has been especially useful in experiments with provers that communicate at runtime within the ILF system [DGHW97]. In order to save time during the communication it was necessary to use the same type encoding for all provers. When working with a single prover, better results may be achieved by taking specific properties of the prover or the actual proof problem into account. The ideal solution might be an automated prover having unification implemented in an exchangeable module, so that various type checking algorithms can be used.

For example the SPASS prover treats unary predicates as types (see [GaMeWe97]) and tries to detect type clashes early. Hence, for this particular prover, the naive interpretation which translates quantification of typed variables into quantification relativized by type predicates, is most efficient – at least for the monomorphic case. For SPASS, the theory $\Theta_{I,\sigma}$ contains axioms describing the declaration of functor symbols and for each type an axiom saying that it is not empty.

[Mel88] proposes a method for encoding monomorphic types that translates also checking the subtype relation into unification problems. This method introduces new variables into the terms. These auxiliary variables in the first order clauses can lead to a larger search space, unless they are treated in a special way by the automated provers.

The input language of some provers (e. g. 3TaP and ProTeIn) supports the encoding of monomorphic tree-like type systems in clauses. When our method is applied in this situation and the modifications described in the last section are applied, the result will be similar.

In the interpretation given above we have made some simplifications compared with the full MIZAR language. We have not considered the set constructor, which, given a set object a and a formula H, constructs an object representing the set of all $x \in a$ such that $H(x)$. This leads to the phenomenon that terms can contain formulas. Handling this situation requires an additional induction over a countable hierarchy of formulas such that terms of level $n+1$ can contain formulas of level n.

We also did not consider types that are defined as classes of structures. This is not a severe restriction since the MIZAR constructors and selectors of structure classes can be translated into ordinary functors. Overloading of functors poses a serious limitation to the interpretation given here. In MIZAR it occurs in the form of redefinitions of value types. They redeclare the value type of a term depending on the types of the arguments. These redeclarations cannot be encoded into the term structure such that type checking

is performed during unification in first order theorem provers. The reason is, that unification works without backtracking from terms to their arguments. It cannot correct a unification clash when it discovers that arguments carry encodings of a more specific type. Nevertheless, it is possible to express overloading by first order formulas so that it can be handled by deductive means.

Recently, Christoph Wernhard has extracted 47 new proof problems for first order provers from an article in the MIZAR library with the ILF system. Unlike earlier test suites, these problems make use of a polymorphic type constructor. The proof problems can be downloaded from the following URL:

`www-irm.mathematik.hu-berlin.de/~ilf/miz2atp/download.html`

These problems use the naive translation mentioned above. They are formulated such, that the involved type information can be easily recovered. Authors of theorem provers are encouraged to modify their provers in order to make efficient use of the type information contained in the problems. The interpretation given above intends to be just one step into this direction.

References

[Ba74] K. J. Barwise: Axioms for abstract model theory; Ann. Math. Logic vol. 7 (1974), 221–265

[DahWer97] I. Dahn, C. Wernhard: First Order Proof Problems Extracted from an Article in the MIZAR Mathematical Library. RISC-Linz Report Series, No. 97-50, pp. 58–62, Johannes Kepler Universität Linz, 1997.

[DGHW97] B. I. Dahn, J. Gehne, Th. Honigmann, A. Wolf: Integration of Automated and Interactive Theorem Proving in ILF. In Proc. CADE-14, pp. 57-60, Springer, 1997.

[GaMeWe97] H. Ganzinger, C. Meyer, C. Weidenbach: Soft Typing for Ordered Resolution. In Proc. CADE-14, pp. 321-335, Springer, 1997.

[Mel88] Mellish, C. S.: Implementing Systemic Classification by Unification. Comp. Ling. 14, 1988, pp 40 – 51

[Rud92] P. Rudnicki: An Overview of the Mizar Project. Proceedings of the 1992 Workshop on Types for Proofs and Programs, Chalmers University of Technology, Bastad 1992.

[Try93] A. Trybulec: Some Features of the Mizar Language, ESPRIT Workshop, Torino 1993.

[WiGe97] G. Wiederhold, M. Genesereth: The Basis for Mediation. To appear IEEE Expert