

Planning Equivalence Proofs

Lassaad Cheikhrouhou and Volker Sorge

Fachbereich Informatik
Universität des Saarlandes
{lassaad|sorge}@cs.uni-sb.de

Abstract

Different axiomatizations of mathematical concepts prove to be useful in a mathematical knowledge base, since each axiomatization of a concept is more or less helpful for the task at hand. To keep the knowledge base consistent, the equivalence of distinct definitions for some concept must be formally proven.

Especially in algebra, where various axiomatizations of an algebraic structure often occur, the proofs of equivalent definitions are canonical in major simplification steps. Starting out with a proof for the equivalence of two distinct definitions for the group concept, we report in this paper on the first attempts of proof planning the equivalence of different definitions for algebraic structures in the Ω MEGA system. We present a proof method that implements the common simplification steps of such theorems and give some relevant methods for closing the subgoals of this method.

1 Introduction

For many proofs of mathematical theorems it is often desirable to choose an appropriate axiomatization for an occurring concept that meets the requirements of the specific problem formulation. To provide this opportunity within a theorem proving environment it is necessary to supply knowledge bases which can contain several parallel axiomatizations for the same mathematical concept. In order to ensure consistency of these kind of knowledge bases it is necessary to provide means for checking the equivalence of given definitions of the same entity. Those problems arise in various domains such as algebra and computation theory. Especially in algebra the proofs of equivalence for parallel definitions can be done in a very canonical way and consist of similar major steps.

In this paper we report on the first attempts of planning such equivalence proofs in the Ω MEGA mathematical assistance system [BCF⁺97]. We emphasize on developing a mechanism that automates the canonical proof scheme for algebraic structures. However, this canonical proof scheme does not correspond to a well-known mathematical proof technique as induction or diagonalization.

In particular we will discuss a systematic way how to construct proofs of equivalence of different axiomatizations of the same algebraic structure in a proof planning framework. We present a well-known example of such a proof from group theory as background for the extraction of the abstract general proof scheme for various algebraic structures. This general scheme is represented in a strategic method that splits the problem into subproblems which can be partially tackled by regular planning methods. We assume that some of these regular

methods work analogously in different algebraic domains and some have to be domain specific. We discuss a few of those methods, both general and typical for group theory.

An important point in the proof of equivalent definitions is the setting of some concepts typical to the one definition, like the operators, some distinguished elements and functions, in termini of the concepts typical to the other definition. We sketch some first ideas how to construct those terms in an appropriate manner using middle-out-reasoning (MOR) [KBB93] methods. Especially with regard to the application of MOR the work reported here can be seen as test bed for finding a balanced and well-working mixture between proof planning, the use of automated reasoners and user interaction.

2 Example: Definitions of Group

To give an impression of the class of problems we are concerned with we give two examples of different yet equivalent definitions for a group. The first one is a standard axiomatization that can be found in most text books.

Definition 1 (Group-1): Let G be a non-empty set, then G together with binary operation ‘ \cdot ’ is a group if the following properties hold:

- G1) For all $a, b \in G$ exists $c \in G$ with $a \cdot b = c$.
- G2) For all $a, b, c \in G$ holds $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- G3) There exists a $e \in G$ such that for all $a \in G$ holds $e \cdot a = a$ and $a \cdot e = a$.
- G4) There exists a unary function $^{-1}$ such that for all $a \in G$ holds $a \cdot a^{-1} = e$ and $a^{-1} \cdot a = e$.

Another possible definition of a group is obtained by expressing the given operation differently without the explicit use of the concept of associativity.

Definition 2 (Group-2): Let G be a non-empty set, then G together with binary operation ‘ $/$ ’ is a group if the following properties hold:

- H1) For all ordered pairs (a, b) where $a, b \in G$ exists a $c \in G$ with $a/b = c$.
- H2) For all $a, b \in G$ holds $a/a = b/b$.
- H3) For all $a, b \in G$ holds $a/(b/b) = a$.
- H4) For all $a, b, c \in G$ holds $(a/a)/(b/c) = c/b$.
- H5) For all $a, b, c \in G$ holds $(a/c)/(b/c) = a/b$.

In the remainder of this section we formalize definitions 1 and 2, and describe the formal proof that both definitions are equivalent¹. We will use a higher order logic based on classical type theory [Chu40] as introduced in [And86], and define the set of base types as the set consisting of the type of truth values o and the type of individuals ι . We then let a set G be of type $\iota \rightarrow o$ and let each element of G be of type ι . Hence a binary operation on G is of type $(\iota, \iota) \rightarrow \iota$ and the property of an e_ι being element of the set G can be modeled as the univariate predicate $G(e)$. We formalize proofs in natural deduction calculus [Gen35] and denote them in the linearized style also introduced in [And86]. In order to preserve readability of the formulas we will, wherever possible, omit the type information and introduce an additional notational convention by writing variables in capital letters and constants in small letters.

We formalize the theorem that definitions 1 and 2 are equivalent straightforward as

¹The complete informal proof is stated in [Hal59] together with a variety of other definitions for the concept of group.

$$\mathbf{T}. \quad \vdash \forall G. [\exists Op_1. \mathbf{group-1}(G, Op_1) \Leftrightarrow \exists Op_2. \mathbf{group-2}(G, Op_2)],$$

were **group-1** and **group-2** are abbreviations for the actual definitions of the groups. The definitions themselves are represented as λ -abstractions and associated to the respective predicate using a special definition operator ‘ $=_{def}$ ’. In detail the definitions are formalized as

$$\begin{aligned} \mathbf{D1.} \quad \mathbf{group-1} &=_{def} \lambda G. \lambda Op_{(\iota, \iota) \rightarrow \iota}. \mathbf{not-empty}(G) \wedge \mathbf{closed-under}(G, Op) \wedge \\ &\quad \mathbf{associative}(G, Op) \wedge \exists E. [G(E) \wedge \mathbf{unit}(G, Op, E) \wedge \\ &\quad \exists I_{\iota \rightarrow \iota}. [\mathbf{inverse-func}(G, Op, E, I)]] \\ \mathbf{D2.} \quad \mathbf{group-2} &=_{def} \lambda G. \lambda Op_{(\iota, \iota) \rightarrow \iota}. \mathbf{not-empty}(G) \wedge \mathbf{closed-under}(G, Op) \wedge \\ &\quad \mathbf{quasi-unit}(G, Op) \wedge \mathbf{quasi-right-unit}(G, Op) \\ &\quad \mathbf{reciprocal}(G, Op) \wedge \mathbf{cancellation}(G, Op) \end{aligned}$$

Each axiom of the respective group definitions is yet again represented as predicates and not as compound propositions. This enables the recognition of similar concepts in different definitions on an abstract level. However, the predicates are once more defined as λ -abstraction and can be introduced in the proof when necessary. For instance, the concept of non-emptiness **not-empty** is defined as the λ -term $\lambda G. \exists X. G(X)$.

Considering theorem **T** we illustrate the typical major steps in proofs of equivalence of group definitions.

1. Split the equivalence
2. Expand the group-definitions into single subgoals
3. Prove the resulting subgoals

We examine the above points step by step and point out the main difficulties for the automation of the proof. Step 1 is simple as it only involves the elimination of the universally quantified variable G and the splitting of the equivalence. Assuming we instantiate g for the variable G we get the two new subgoals

$$\begin{aligned} \mathbf{S1.} \quad &\exists Op_1. \mathbf{group-1}(g, Op_1) \vdash \exists Op_2. \mathbf{group-2}(g, Op_2), \\ \mathbf{S2.} \quad &\exists Op_2. \mathbf{group-2}(g, Op_2) \vdash \exists Op_1. \mathbf{group-1}(g, Op_1), \end{aligned}$$

to which the steps 2 and 3 will be applied in turn.

In step 2 the abstract predicates **group-1** and **group-2** are replaced by their respective definitions and the resulting formulas simplified by interwoven elimination of existentially quantified variables and splitting of conjunctions. This procedure can at first be applied to the antecedents of the two subgoals only. Here each of the existentially quantified variables can directly be instantiated with simple witness terms and any resulting conjunction can be split. When expanding the succedents, however, the elimination of existential quantifiers is less trivial as possible instantiations for variables generally depend on the witness terms introduced beforehand.

We first consider subgoal **S1** and assume that, when expanding the left hand side of the derivation, the occurring variables Op_1 , E , and I are instantiated by the constants \cdot , e , and $^{-1}$ respectively. The set **H** in Figure 1 displays all hypotheses obtained after splitting all conjunctions in the instantiated definition. In order to perform the similar expansion step on the right hand side of the subgoal the variable Op_2 has to be instantiated with a λ -term depending on both \cdot and $^{-1}$, namely $\lambda XY. (X \cdot Y^{-1})$. This expansion step leads then to the open subgoals (1)–(6) displayed in Figure 1 which are then proved separately in step 3.

The first subgoal (1) can be directly closed by the first hypothesis, whereas the remaining subgoals are further expanded using the respective definitions. The available hypotheses are

$\mathbf{H} \vdash \text{not-empty}(g)$	(1)
$\mathbf{H} \vdash \text{closed-under}(g, \lambda XY.X \cdot Y^{-1})$	(2)
$\mathbf{H} \vdash \text{quasi-unit}(g, \lambda XY.X \cdot Y^{-1})$	(3)
$\mathbf{H} \vdash \text{quasi-right-unit}(g, \lambda XY.X \cdot Y^{-1})$	(4)
$\mathbf{H} \vdash \text{reciprocal}(g, \lambda XY.X \cdot Y^{-1})$	(5)
$\mathbf{H} \vdash \text{cancellation}(g, \lambda XY.X \cdot Y^{-1})$	(6)
$\mathbf{H} = \{ \text{not-empty}(g), \text{closed-under}(g, \cdot), \text{associative}(g, \cdot),$	
$g(e), \text{unit}(g, \cdot, e), \text{inverse-func}(g, \cdot, e, ^{-1}) \}$	

Figure 1: Subgoals for proof of example 2

expanded as well in order to close the resulted subgoals. Hereby, we employ besides basic simplification tactics, as the splitting of a conjunction subgoal to its conjuncts, tactics based on equality inferences. For instance, subgoal (3) in Figure 1 is expanded to the subgoal $[\forall A, B. (g(A) \wedge g(B)) \Rightarrow A \cdot A^{-1} = B \cdot B^{-1}]$ which is reduced by simplification tactics to the equality $[a \cdot a^{-1} = b \cdot b^{-1}]$. Both sides of this equality are reduced to the same constant e employing equalities that result from the expansion of hypothesis $\text{inverse-func}(g, \cdot, e, ^{-1})$ to the formula $[\forall X. g(X) \Rightarrow (g(X^{-1}) \wedge X^{-1} \cdot X = e \wedge X \cdot X^{-1} = e)]$.

For the proof of subgoal **S2** three different existentially quantified variables need to be instantiated with term depending on the operator Op_2 . Assuming $/$ is introduced as a witness term for Op_2 , the necessary term to instantiate Op_1 is $\lambda XY.X / ((Y/Y)/Y)$. The instantiated goal and the new hypothesis are expanded using the associated definitions and are subsequently split. We obtain among others the subgoal

$$\exists E. [g(E) \wedge \text{unit}(g, \lambda XY.X / ((Y/Y)/Y), E)] \wedge \exists I. \text{inverse-func}(g, \lambda XY.X / ((Y/Y)/Y), E, I)$$

Here appropriate terms must be given for the variables E and I . While the inverse function I corresponds to the λ -term $\lambda X.(X/X)/X$ which depends on $/$ only, the unit element E depends additionally on some element of g . The fitting term can be derived using the definitions of $\text{not-empty}(g)$ and $\text{closed-under}(g, \cdot)$ and corresponds to (X'/X') where X' is an arbitrary free variable.

The rest of the proof for **S2** can then be accomplished similarly to the one of **S1** by expansion and application of both basic simplification tactics and tactics based on equality inferences.

3 A General Planning Strategy

In this section we generalize the equivalence proof presented above to a methodology that can be used in the proof planning framework of Ω MEGA [BCF⁺97] to construct similar proofs for equivalence of definitions of various algebraic structures. After giving an overview on the proof planning framework of Ω MEGA, we outline a strategic method that implements the first simplifying steps in such proofs and we describe some (domain) specific methods that should be used to solve the resulting subgoals.

3.1 Proof Planning in Ω MEGA

The Ω MEGA system is an interactive proof development environment where proofs can be constructed with the support of various automatic tools, such as a proof planner, automated theorem provers, etc. The central data structure in Ω MEGA is the so-called *proof plan data structure* (\mathcal{PDS}) which represents a partial proof of the theorem to be shown at several levels of abstraction. The levels correspond to different abstractions of the formal ND (natural deduction) proof. A \mathcal{PDS} level is a proof tree with the theorem as its root and the hypotheses as its leaves. The child nodes in such a proof tree are either the premises for the justification of their *closed* parent node or the *supports* for their *open* parent node. The supports of an *open* node consist of a heuristically determined subset of the hypotheses of this node and of their deduced consequences, which should be used to close this open node.

A justification of a closed node denotes that the conjecture of this node follows logically by the stated *inference method* from the conjectures of the premises. “*Inference method*” is the common designation of basic calculus inference rules (ND rules) and coherent compositions of these rules. A *non-basic inference method* can be a tactic, a proof schema that represents a proof idea (technique) with still some gaps, or a call to an external reasoner whose returned result can be transformed into a ND proof. Non-basic inference methods abstract many individual inference steps by grouping them into one proof step at the next higher level of abstraction.

An initial \mathcal{PDS} consists of an open node, the theorem, and its supports, the hypotheses. A formal ND proof of this problem is constructed by the application of inference methods to open nodes and the expansion of non-basic inference methods. This construction process can be done interactively and/or with the help of a planner which uses the inference methods as plan operators. Plan operators, designated as *methods*, are actually specifications that represent applicability condition and application effects of inference methods. In Ω MEGA, we use a declarative representation as much as possible for methods, such that meta-methods can be employed to reformulate methods in analogy-driven theorem proving [Mel98]. A method is represented as a frame-like structure, the meaning of the slots in Figure 2 will be explained in the next section where we describe the method **Equiv-Structure**.

3.2 The Equiv-Structure Method

We first present the main considerations leading to a strategic method that is applicable for a variety of algebraic structures by generalizing the equivalence statements for the definitions of groups in section 2 and thereby gaining a generic scheme for the theorems stating the equivalence between two different definitions. The general theorem can be formulated as $\forall \mathcal{S}. \exists \mathcal{O}_1. \text{struct}_1(\mathcal{S}, \mathcal{O}_1) \Leftrightarrow \exists \mathcal{O}_2. \text{struct}_2(\mathcal{S}, \mathcal{O}_2)$ where calligraphic letters stand for finitely many terms. The respective definitions of the considered algebraic structure are designated by struct_1 and struct_2 . The involved sets are represented by \mathcal{S} ; \mathcal{O}_1 and \mathcal{O}_2 correspond to the operations defined on these sets. As an instance of this general theorem scheme one can consider the theorem **T** from section 2. As examples of algebraic structures including more than one operator or more than one set of individuals one could think of definitions for *fields* or *modules*.

The first major steps for proving such a generic equivalence theorem are implemented in the strategic method **Equiv-Structure** in figure 2. The *proof schema* of **Equiv-Structure** represents the proof part that should be inserted into a \mathcal{PDS} when expanding this method. Only the boundaries of the proof schema, i.e., the method *conclusions* and the method

Method : Equiv-Structure	
Conclusions	$(\ominus 16)$
Premises	$(\oplus \mathcal{G}_1) (\oplus \mathcal{G}_2)$
Computation	$S' \leftarrow \text{Var2Const}(\mathcal{S}),$ $\mathcal{O}'_1 \leftarrow \text{Var2Const}(\mathcal{O}_1), \mathcal{Q}'_2 \leftarrow \text{Var2Mvar}(\mathcal{O}_2),$ $\Psi_{S', \mathcal{O}'_1} \leftarrow \text{ApplyDefn}(\text{struct}_1, \langle S', \mathcal{O}'_1 \rangle), \mathcal{L}_1 \leftarrow \text{AndExistsE}*(\Psi_{S', \mathcal{O}'_1})$ $\Phi_{S', \mathcal{Q}'_2} \leftarrow \text{ApplyDefn}(\text{struct}_2, \langle S', \mathcal{Q}'_2 \rangle), \langle \mathcal{G}_1, \underline{\mathcal{M}}_1 \rangle \leftarrow \text{AndExistsI}*(\Phi_{S', \mathcal{Q}'_2})$ $\mathcal{O}'_2 \leftarrow \text{Var2Const}(\mathcal{O}_2), \mathcal{Q}'_1 \leftarrow \text{Var2Mvar}(\mathcal{O}_1),$ $\Pi_{S', \mathcal{O}'_2} \leftarrow \text{ApplyDefn}(\text{struct}_2, \langle S', \mathcal{O}'_2 \rangle), \mathcal{L}_2 \leftarrow \text{AndExistsE}*(\Pi_{S', \mathcal{O}'_2})$ $\Xi_{S', \mathcal{Q}'_1} \leftarrow \text{ApplyDefn}(\text{struct}_1, \langle S', \mathcal{Q}'_1 \rangle), \langle \mathcal{G}_2, \underline{\mathcal{M}}_2 \rangle \leftarrow \text{AndExistsI}*(\Xi_{S', \mathcal{Q}'_1})$
Actions	$\ll \mathcal{G}_1 : +(\mathcal{L}_1), \Leftrightarrow(8, 10) \gg \ll \mathcal{G}_2 : +(\mathcal{L}_2), \Leftrightarrow(1, 3) \gg$
Proof Schema	<ol style="list-style-type: none"> 1. $\vdash \text{struct}_2(S', \mathcal{O}'_2)$ (Hyp) 2. 1 $\vdash \Pi_{S', \mathcal{O}'_2}$ (DefnE(struct₂) 1) 3. $\vdash \exists \mathcal{O}_2 \bullet \text{struct}_2(S', \mathcal{O}_2)$ (Hyp) 4. 1 $\vdash \Xi_{S', \mathcal{Q}'_1}$ (AndExistsI*($\underline{\mathcal{M}}_2$) \mathcal{G}_2) 5. 1 $\vdash \text{struct}_1(S', \mathcal{Q}'_1)$ (DefnI(struct₁) 4) 6. 1 $\vdash \exists \mathcal{O}_1 \bullet \text{struct}_1(S', \mathcal{O}_1)$ (ExistsI*(\mathcal{Q}'_1) 5) 7. 3 $\vdash \exists \mathcal{O}_1 \bullet \text{struct}_1(S', \mathcal{O}_1)$ (ExistsE*(\mathcal{O}'_2) 3 6) 8. $\vdash \text{struct}_1(S', \mathcal{O}'_1)$ (Hyp) 9. 8 $\vdash \Psi_{S', \mathcal{O}'_1}$ (DefnE(struct₁) 8) 10. $\vdash \exists \mathcal{O}_1 \bullet \text{struct}_1(S', \mathcal{O}_1)$ (Hyp) 11. 8 $\vdash \Phi_{S', \mathcal{Q}'_2}$ (AndExistsI*($\underline{\mathcal{M}}_1$) \mathcal{G}_1) 12. 8 $\vdash \text{struct}_2(S', \mathcal{Q}'_2)$ (DefnI(struct₂) 11) 13. 8 $\vdash \exists \mathcal{O}_2 \bullet \text{struct}_2(S', \mathcal{O}_2)$ (ExistsI*(\mathcal{Q}'_2) 12) 14. 10 $\vdash \exists \mathcal{O}_2 \bullet \text{struct}_2(S', \mathcal{O}_2)$ (ExistsE*(\mathcal{O}'_1) 10 13) 15. $\vdash \exists \mathcal{O}_1 \bullet \text{struct}_1(S', \mathcal{O}_1) \Leftrightarrow \exists \mathcal{O}_2 \bullet \text{struct}_2(S', \mathcal{O}_2)$ (EquivSplit 14 7) 16. $\vdash \forall \mathcal{S} \bullet \exists \mathcal{O}_1 \bullet \text{struct}_1(\mathcal{S}, \mathcal{O}_1) \Leftrightarrow \exists \mathcal{O}_2 \bullet \text{struct}_2(\mathcal{S}, \mathcal{O}_2)$ (ForallI*(S') 15)

Figure 2: The **Equiv-Structure** Method

premises are considered in the application of the associated method.

The *conclusions* of **Equiv-Structure** consist of an annotated method node that corresponds to the ND line closed by this method. The annotation \ominus specifies that the associated ND line is reduced to new open ND lines. Thus, the **Equiv-Structure** method can be applied to goals whose formulas match the formula of line 16.

The *premises* of **Equiv-Structure** consist of two annotated method variables that represent list of ND lines, from which the conclusions logically follow. The associated ND lines to premises annotated by \oplus are added as open nodes to the \mathcal{PDS} and can be regarded as the subgoals of the method. In contrast to the conclusions of the **Equiv-Structure** method, we have to use method variables instead of method nodes to represent the premises, since the latter can not be specified declaratively. The slot *computation* comprises the function calls needed for determining the method subgoals.

Generally, the supports of each method subgoal correspond to those of the considered goal and to the new hypotheses inserted by this method. Moreover, the support list of a subgoal can be affected by the method's *actions* which specify which ND lines have to be eliminated from this list and which lines must be added.

When the conclusion line of the **Equiv-Structure** method (line 16) is successfully matched with an open ND line in the current \mathcal{PDS} , the method variables \mathcal{S} , \mathcal{O}_1 , struct_1 , \mathcal{O}_2 , and struct_2

are simultaneously bound to appropriate object terms. Using these bindings, the computation instructions of **Equiv-Structure** determine the new ND lines to be associated to the method variables \mathcal{G}_1 , \mathcal{L}_1 , \mathcal{G}_2 , and \mathcal{L}_2 . Considering the example in section 2, \mathcal{L}_1 would be exactly the set **H** in Figure 1 and \mathcal{G}_1 consists of the subgoals (1)–(6) displayed in the same Figure with a meta-variable \underline{Op}_2 instead of the λ -term $\lambda XY.(X \cdot Y^{-1})$.

Without invoking the tactics in the proof schema of **Equiv-Structure**, the computation functions allow to determine the method subgoals depending on the given goal. In the sequel of this section, we outline these computation instructions and discuss how they are related to the tactics in the proof schema.

Split the equivalence: The function *Var2Const* computes for the variables bound to \mathcal{S} a list of new constants \mathcal{S}' to be employed as a parameter of the tactic **ForallI*** which justifies line 16 from line 15. The latter line is justified by the tactic **EquivSplit** which splits the rest of the proof schema into two symmetric proof trees eventually leading to the two lists of subgoals \mathcal{G}_1 and \mathcal{G}_2 . Considering the theorem **T** in section 2, the introduced constant g for the universally quantified variable G is the sole element of \mathcal{S}' and the subgoals **S1** and **S2** correspond to the lines 14 and 7 in the proof schema of **Equiv-Structure**.

As the computation instructions for both subgoal lists \mathcal{G}_1 and \mathcal{G}_2 are similar, we concentrate on the first one and describe how it is incrementally determined by the corresponding function calls in the *computation* slot.

Introduce witness terms: The tactic **ExistsE*** is simulated by the generation of new constants \mathcal{O}'_1 for the existential quantified variables in \mathcal{O}_1 . This tactic proves line 14 using the hypothesis 10 introduced by **EquivSplit** and the line 13 as premises. The latter premise follows from line 12 by the tactic **ExistsI*** whose application is imitated by generating new meta-variables \mathcal{O}'_2 ² as place holder for the witness terms of the existential quantified variables in \mathcal{O}_2 . Since these witness terms can not be determined at this point in the proof, we use meta-variables instead which have to be progressively instantiated while closing schematic subgoals that are represented by formulas containing meta-variables. In the example of section 2, \mathcal{O}'_1 would consist of the constant \cdot introduced for the existentially variable Op_1 , and the list \mathcal{O}'_2 has to contain a meta-variable \underline{Op}_2 which is finally instantiated to the λ -term $\lambda XY.(X \cdot Y^{-1})$.

We expect that the appropriate instantiations of meta-variables can be mostly done by the application of middle-out-reasoning (MOR) [KBB93] methods which heuristically reduce schematic goals and restrict the bindings of occurring meta-variables with some constraints. Thereby it should be possible to either directly determine the right instantiations or to use one of Ω MEGAS other reasoning components, such as an automated theorem prover or a constraint solver, to compute the desired term. However, it might be necessary that very tricky instantiations must be provided interactively by the user.

Expand the Structure definitions: The next calls of the function *ApplyDefn* substitute the predicates struct_1 and struct_2 with their respective properties. Considering the example in section 2, this function would return the β -normalform of the application of the λ -abstractions in the definitions **D1** and **D2** to the pairs (g, \cdot) , and (g, \underline{Op}_2) , respectively.

The resulting formulas $\Psi_{\mathcal{S}', \mathcal{O}'_1}$ and $\Phi_{\mathcal{S}', \mathcal{O}'_2}$ are mainly conjunctions of predicates with some existentially quantified variables representing certain elements of the occurring sets (compare definitions **D1** and **D2**, section 2). But as the positions and therefore the scopes of the

²Underlined letters denote (lists of) meta-variables.

existential quantifiers are a priori undetermined, a common representation of such formulas can only be approximately described. Let P_1, \dots, P_k be the properties of structure struct_1 . Then we can view the formula $\Psi_{\mathcal{S}', \mathcal{O}'_1}$ as

$$\exists E_1 \dots E_n \bullet s'_1(E_{j_1}) \wedge \dots \wedge s'_m(E_{j_m}) \wedge P_1^{\mathcal{O}'_1, \mathcal{E}} \wedge \dots \wedge P_k^{\mathcal{O}'_1, \mathcal{E}} \quad (7)$$

where s'_1, \dots, s'_m are, not necessarily distinct, elements of the set list \mathcal{S}' and $\{E_{j_1}, \dots, E_{j_m}\}$ is a subset of $\{E_1, \dots, E_n\}$. Furthermore the existential quantifications do not need to be at the beginning of the formula but can be intermixed with the conjunctions and the P_i can depend on the operations \mathcal{O}'_1 and some of the E_j which is indicated by superscript letters.

Although it is not possible to exactly determine (7), the simplification of this formula can be clearly defined as successive applications of \wedge - and \exists -eliminations. The latter ones are performed by using new distinct constants e'_j for the variables E_j . The function *AndExistsE** carries out this simplification and returns the list \mathcal{L}_1 of ND lines with the formulas $s'_1(e'_{j_1}), \dots, s'_m(e'_{j_m}), P_1^{\mathcal{O}'_1, \mathcal{E}'}, \dots$, and $P_k^{\mathcal{O}'_1, \mathcal{E}'}$. For instance, this function introduces in the example of section 2 the constants e and $^{-1}$ for the existentially quantified variables E , and I in λ -abstraction defining **group-1** and returns the supports in set **H** of Figure 1.

The elements of \mathcal{L}_1 are added as supports for the open nodes in \mathcal{G}_1 which are determined by the call of function *AndExistsI** on $\Phi_{\mathcal{S}', \mathcal{O}'_2}$. This function imitates the effects of the tactic **AndExistsI*** by employing \wedge - and \exists -introduction rules to the given formula, producing a set of new open lines. Moreover the witness terms introduced during the \exists -introduction steps are again meta-variables, as we usually can not yet determine the required witness terms.

The open ND lines in \mathcal{G}_2 and their supports in \mathcal{L}_2 are determined in the same way by the specified function calls. The resulted open ND lines are then inserted as subgoals of the **Equiv-Structure** method. In addition to the actions-slot of this method, more control information is represented by control rules which prefer some specific methods while closing these subgoals. The evaluation of these control rules is restricted to the *scope* of the **Equiv-Structure** method, where a method *scope* corresponds to the subproof tree in the *PDS* its root is justified by this method.

3.3 Some Specific Methods

In many cases, the axioms of equivalent definitions of algebraic structures are somehow related to each other. Some axioms of the one definition can have similar, weaker, or stronger axioms of the other definition. Therefore, we employ the methods **same**, **weaken-axiom**, and **strengthen-axiom** as some specific methods to be used while proving the subgoals delivered by the strategic method **Equiv-Structure**:

- The method **same** closes a goal from a support line with the same formula.
- The goal of the method **weaken-axiom** is represented by the formula $\bigwedge_{i=0}^n P_i \Rightarrow \bigwedge_{j=0}^m C_j$. An open goal matching this formula can be closed by the method, when a support line of the form $\bigwedge_{i=0}^k P'_i \Rightarrow \bigwedge_{j=0}^l C'_j$ is available, for which $\{P'_0, \dots, P'_k\}$ is a subset of $\{P_0, \dots, P_n\}$ and the set of conclusions $\{C_0, \dots, C_m\}$ is a subset of $\{C'_0, \dots, C'_l\}$.
- The method **strengthen-axiom** can be used to reduce a goal that matches the formula $\bigwedge_{i=0}^n P_i \Rightarrow \bigwedge_{j=0}^m C_j$ provided a support line matching $\bigwedge_{i=0}^k P'_i \Rightarrow \bigwedge_{j=0}^l C'_j$ is given, where the set of premises $\{P'_0, \dots, P'_k\}$ is a subset of $\{P_0, \dots, P_n\}$ and the set $\{C'_0, \dots, C'_l\}$ is a subset of $\{C_0, \dots, C_m\}$. Each conjunct C_j that does not belong to the set of conclusions $\{C'_0, \dots, C'_l\}$ is considered as a subgoal to be proven with the additional supports $P_0, \dots, P_n, C'_0, \dots, C'_{l-1}$, and C'_l .

As the strategic method **Equiv-Structure** can deliver schematic goals, we need specific methods for MOR which possibly introduce (binding) constraints for the occurring meta-variables. Each of the above methods can be generalized in order to be applicable in a flexible setting, where goals are represented with the help of meta-variables. However, the generalization must be carried out carefully to restrict the application of such methods for sensible situations. For instance, the method **same-MOR** is a generalization of **same** which closes a goal $\varphi(\underline{\mathcal{M}})$ using a hypothesis $\varphi(\underline{\mathcal{T}})$ and binding the meta-variables in $\underline{\mathcal{M}}$ to the associated terms in $\underline{\mathcal{T}}$. An additional application condition of this method demands that φ must be matched to a fully instantiated object term, e.g., a predicate symbol.

3.4 Some Domain Specific Methods

In the following we present two typical methods for group theory which are significant in solving the kind of subgoals that can be delivered by the strategic method **Equiv-Structure** applied in a context involving groups. These methods somehow imitate the problem solving behavior of mathematicians. Determining the unit (inverse) element in a group is usually achieved by first finding a left unit (inverse) element and a right unit (inverse) element, and then stating the equality of both elements. This is implemented by the method **unit2lr** (**inverse2lr**):

- The method **unit2lr** should be applied to subgoals resulted by expanding the concept **unit**, i.e., to formulas of the form $[g(\underline{E}) \wedge \forall X. g(X) \Rightarrow (\underline{Op}(\underline{E}, X) = X \wedge \underline{Op}(X, \underline{E}) = X)]$, where \underline{E} and \underline{Op} represent the unit element and the operator respectively and can be matched with schematic terms. The application of this method delivers two subgoals: one for the property of a left unit element $[g(\underline{E}) \wedge \forall X. g(X) \Rightarrow \underline{Op}(\underline{E}, X) = X]$, and a second one for the property of a right unit element $[g(\underline{E}_r) \wedge \forall X. g(X) \Rightarrow \underline{Op}(X, \underline{E}_r) = X]$. The equality of both unit elements $[\underline{E} = \underline{E}_r]$ is proven inside the tactical content of the method using the property of the left unit element.
- The goal of the method **inverse2lr** is a conjunction $[g(\underline{Y}) \wedge \underline{Op}(\underline{Y}, c) = \underline{E} \wedge \underline{Op}(c, \underline{Y}) = \underline{E}]$ stating that some element \underline{Y} is the inverse of an element c , where the meta-variables \underline{Y} , \underline{Op} , and \underline{E} can be matched with schematic terms. The application of this method deliver the subgoals $[g(\underline{Y})]$, $[\underline{Op}(\underline{Y}, c) = \underline{E}]$, $[\underline{Op}(c, \underline{Y}_r) = \underline{E}]$, and $[\underline{Y} = \underline{Y}_r]$, where \underline{Y}_r stands for a right inverse element of the element c .

Furthermore, we speculate that some other domain specific methods can be defined which are relevant for proving equalities in the context of groups. More examples must be investigated to determine such kind of methods.

4 Conclusion and Future Work

Many mathematical concepts can be described by different axiomatizations which can prove useful for the task at hand. It is therefore appropriate to specify a concept by several equivalent definitions in a mathematical knowledge base. To keep the knowledge base consistent, the equivalence of distinct definitions for some concept must be formally proven. Especially in algebra, where various axiomatizations of an algebraic structure often occur, the proofs of equivalent definitions are canonical in major steps.

We have presented the first steps towards an automation of the planning process of equivalence proofs for different definitions of algebraic structures. The major proof steps are

constructed with the strategic method **Equiv-Structure** that is generic enough to handle various algebraic entities. Some more specific methods are then employed to proof single axioms from the given ones. We use middle-out-reasoning to postpone the selection of crucial terms to some point in the proof where more information on the shape of the required terms is available. So far we have tested the approach for several equivalent definitions of groups and we want to test whether the approach is general enough by considering definitions for more complex algebraic structures such as modules.

In some related work [Kun92, McC93] equivalence proofs have been automatically constructed in order to show the existence of single axioms defining groups. But the operations on groups have been explicitly formalized as axioms for the proofs, that is single axioms have been formulated with respect to the functions (for example multiplication and inversion) of the original axiomatization. Moreover the postulation that a group is a non-empty set, secluded with respect to the binary operation has implicitly been assumed.

Our work attempts to assist proving equivalence between axiomatizations in a more general setting by combining general and domain specific knowledge. A proof planning framework like Ω MEGA's, where the object language is a sorted higher order logic with partial functions, allows for a more natural representation of mathematical knowledge and consequently more intuitive proof construction.

So far experiments with the higher order automatic theorem prover TPS [ABI⁺96] have shown that even simpler versions of the definitions presented in this paper are beyond the scope of current higher order theorem provers. Here proof planning has the advantage that domain specific knowledge can be used to guide the proof search. However, we believe that when carefully employed to prove certain subgoals TPS can not only provide subproofs but also useful instantiations for occurring meta-variables. Currently we are investigating the use of TPS during the middle-out-reasoning process and the application of computer algebra for closing equational subgoals.

References

- [ABI⁺96] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A Theorem Proving System for Classical Type Theory. *Journal of Automated Reasoning*, 16(3):321–353, 1996.
- [And86] P. B. Andrews. *An Introduction To Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, San Diego, CA, USA, 1986.
- [BCF⁺97] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. Ω Mega: Towards a Mathematical Assistant. In W. McCune, editor, *Proceedings of the CADE-14*, LNAI, Townsville, Australia, 1997. Springer Verlag, Berlin, Germany.
- [Chu40] A. Church. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic*, 5:56–68, 1940.
- [Gen35] G. Gentzen. Untersuchungen über das Logische Schließen I und II. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935.
- [Hal59] M. Hall. *The Theory of Groups*. The Macmillan Company, New York, USA, 1959.
- [HKRS94] X. Huang, M. Kerber, J. Richts, and A. Sehn. Planning Mathematical Proofs with Methods. *Journal of Information Processing and Cybernetics*, 30(5/6):277–291, 1994.
- [KBB93] I. Kraan, D. Basin, and A. Bundy. Middle-Out Reasoning for Logic Program Synthesis. techreport MPI-I-93-214, Max-Planck-Institut, Saarbrücken, Germany, April 1993.
- [Kun92] K. Kunen. Single Axioms for Groups. *Journal of Automated Reasoning*, 9(3):291–308, 1992.
- [McC93] W. McCune. Single Axioms for Groups and Abelian Groups with Various Operations. *Journal of Automated Reasoning*, 10(1):1–13, 1993.
- [Mel98] E. Melis. The Heine-Borel challenge problem: In honor of Woody Bledsoe. *Journal of Automated Reasoning*, 21, 1998. forthcoming.