Using Functors in Organizing Proofs

Elena Tomuta
Research Institute for Symbolic Computation
A4232 Schloss Hagenberg, Austria
tomuta@RISC.uni−linz.ac.at

Abstract
Functors are the main mechanism for structuring knowledge in the *Theorema* system. In this paper we give an overview of the facilities the system provides for manipulating functors, domains and domain properties and we present an approach to using the structure introduced by functors in guiding proofs. The approach was implemented in a prover specialised in proving statements involving domains and domain properties. The interface and structure of the prover are explained and a sample proof produced by the system is given.

1. Introduction

The *Theorema* system consists in several powerfull specialized provers: the predicate logic prover [Buchberger , Jebelean 1997], the induction provers for lists and natural numbers ([Buchberger 1996b] and [Buchberger, Vasaru 1997] , the simplification prover [Buchberger, Marin 1997]. However, even trivial proofs in "real−life" mathematics involve mixed, natural deduction, simplification and/or induction steps. The succession of these steps often depends (to some degree at least) on the *mathematical domain(s)* in which we prove. We attempt to determine how the structure of the mathematical facts we handle (expressed by domains) can be used to control provers' interaction when building up a proof.
Even for subproofs performed by just one prover, the information contained in the domain structure can be used to greatly reduce the complexity of the proof search space.

Summarizing, we are interested in 2 aspects of using functors to structure proofs:
◇ coordinating specialized provers by using knowledge about the hierarchical structure of the domains involved in the proofs;
◇ using the knowledge about domains as control information which reduces the proof complexity for individual provers.

As a case study we consider the problem of proving "knowledge transport properties", namely, that if certain properties holds in some given domains and we build up a new domain based on the given ones then another property will hold in the new domain.
This class of mathematical statements can be formulated more formally as follows:
We want to prove

(*Functor.Formula*) $(\mathrm{D} = \mathrm{F}[D_1, \ldots, D_k] \wedge P_1[D_1] \wedge \ldots \wedge P_n[D_n]) \Rightarrow \mathrm{P[D]}$

where
F is a functor,
$D, D_1, \ldots, D_n$ are domains,
$P, P_1, \ldots, P_n$ are sets of formulae expressing properties of the respective domains.

The paper is structured as follows: in section 2 we present the functions provided for defining functors and domains and for testing the existence of a domain definition associated with a given symbol. In section 3 we show how properties about functors can be formulated and manipulated. Finally section 4 presents the functor

prover, and explains the ideas behind. An automatically generated proof is given in the Appendix.
The meaning of the notion of functor is presented e.g. in [Buchberger 1996a].

2. Defining Functors and Domains

Functors can be defined by using the function *Functor.* The syntax for functor definitions is the following:

$F[D, D_1, D_2, \ldots D_n] := Functor[\langle localVariables \rangle,$
$LabeledFormula_1;$
$LabeledFormula_2;$
.
.
.
$LabeledFormula_k$
];

where
◇ $D_1, D_2, \ldots, D_n$ are the input domains and D is the output domain;
◇ *localVariables* is a list of (possibly initialized) variables which are local to the scope of the functor definition;
◇ the labeled formulae *1* through *k* are formulae in the Theorema syntax defining the functor's operators.

- **Example(Defining the functor Complexification)**

The functor *Complexification*, which builds the domain *C* of complex numbers over the given domain *A* can be defined as follows:

$$\text{Complexification}[C\_, A\_] :=$$
$$\text{Functor}\Big[\langle\rangle,$$
$$\Big\langle \text{"}\in\text{"},\ \forall_{\langle x\rangle} \underset{C}{\in}[x] \Leftrightarrow \exists_{\langle xr, xi\rangle} \underset{A}{\in}[xr] \wedge \underset{A}{\in}[xi] \wedge (x \equiv \langle xr, xi\rangle) \Big\rangle;$$

$$\Big\langle \text{"}\oplus\text{"},\ \forall_{\langle x1, y1, x2, y2\rangle} \langle x1, y1\rangle \underset{C}{\oplus} \langle x2, y2\rangle \equiv \Big\langle x1 \underset{A}{\oplus} x2,\ y1 \underset{A}{\oplus} y2\Big\rangle\Big\rangle;$$

$$\Big\langle \text{"}\circ\text{"},\ \underset{C}{\circ} \equiv \Big\langle \underset{A}{\circ}, \underset{A}{\circ}\Big\rangle\Big\rangle;$$

$$\Big\langle \text{"inv"},\ \forall_{\langle x, y\rangle} \underset{C}{\text{inv}}[\langle x, y\rangle] \equiv \Big\langle \underset{A}{\text{inv}}[x], \underset{A}{\text{inv}}[y]\Big\rangle\Big\rangle;$$

$$\Big\langle \text{"}\otimes\text{"},\ \forall_{\langle x1, y1, x2, y2\rangle} \langle x1, y1\rangle \underset{C}{\otimes} \langle x2, y2\rangle \equiv$$
$$\Big\langle \Big(x1 \underset{A}{\otimes} x2\Big) \underset{A}{\oplus} \underset{A}{\text{inv}}\Big[\Big(y1 \underset{A}{\otimes} y2\Big)\Big],\ \Big(x1 \underset{A}{\otimes} y2\Big) \underset{A}{\oplus} \Big(x2 \underset{A}{\otimes} y1\Big)\Big\rangle\Big\rangle$$

$$\Big];$$

Domains are defined by means of the function *DefineDomain,* which takes two arguments: a symbol representing a domain to be defined and an expression of the form $F[D_1, D_2, \ldots, D_n]$, where F is a functor and $D_1, D_2, \ldots, D_n$ are the domains passed as arguments.

Thus the syntax for defining a new domain D in terms of the functor F and the domains $D_1, D_2, \ldots, D_n$ is:

$$DefineDomain[\, D,\ F[D_1,\ D_2,\ \ldots,\ D_n]\,];$$

or, in a more intuitive form:

$$D \sim DefineDomain \sim F[D_1,\ D_2,\ \ldots,\ D_n].$$

The second form suggests that *DefineDomain* can be seen as a specialized assignment used in domain definitions in place of the usual assignment forms (:= and ==) of *Mathematica.*
For example having defined the functor *Complexification* we can define the domain *C* as the Complexification of *A* as follows:

```
C ~DefineDomain~ Complexification[A];
```

The function *GetFunctorDefinition* applied to a symbol returns the functor expression associated with a given symbol. If no such definition was made in the current *Mathematica* session, the function returns *Null*.
The function *IsDomain* returns *True* if applied to a symbol for which a functor definition is known and *False* otherwise.
The function *GetDomainDefinition* applied to a domain returns the list of formulae defining the domain as a functor of (possibly) other domains.

3. Defining Domain Properties

Properties of domains can be defined by means of the functions *DefineProperty* and *Assume.* The function *DefineProperty* is used for assigning a name to a set of formulae describing the properties of a domain. Its syntax is as follows:

PropertyName[DomainVariable, Operators] ~DefineProperty~ ListOfFormulae;

where
◇ *ListOfFormulae* is a list of labeled formulae stating properties of the domain refered to by *DomainVariable*
◇ *Operators* is a list of operators with respect to which the properties are defined;

The sequence *DomainVariable, Operators* can be seen as the list of formal parameters to the property definition and the function *~DefineProperty~* as the assignement used for domain properties.
Property definitions are stored in an internal structure of the Theorema system. The function *IsProperty* applied to a symbol returns *True* if the symbol is a property name and false otherwise.
The function *GetDefinition* returns the list of labeled formulae contained in the definition of a property, instantiated with a particular domain symbol and its operators, or in other words, it instantiates the list of formal parameters in the body of the property definition with actual parameters.

To specify that a defined property holds for a given domain the user can use the function *Assume,* with the following syntax:

Assume[PropertyName[ActualDomain, ActualOperators];

where *ActualDomain* is the name of the domain about which we assume that it has property *PropertyName* and *ActualOperators* are operators of the domain *ActualDomain*.
The function *GetDomainProperties* returns the list of properties assumed about a given domain. The function *RemoveDomainProperty* can be used to "forget" a property of a domain.

■ **Example(Defining and Using the Ring Property)**

As an example let us define the ring property for a domain R with respect to the operations: "⊕" (addition), "o" (zero), "inv" (inverse w.r.t. addition) and "⊗" (multiplication).

$$\text{Ring}[R, \text{"}\oplus\text{"}, o, inv, \text{"}\otimes\text{"}] \sim\text{DefineProperty}\sim$$
$$\Big\langle\Big\langle \text{"}\oplus\text{.assoc"}, \ \forall_{\langle x, y, z\rangle} \Big(\underset{R}{\in}[x] \wedge \underset{R}{\in}[y] \wedge \underset{R}{\in}[z]\Big) \Rightarrow ((x \oplus y) \oplus z \equiv x \oplus (y \oplus z))\Big\rangle,$$
$$\Big\langle \text{"}\oplus\text{.o.left"}, \ \forall_{\langle x\rangle} \underset{R}{\in}[x] \Rightarrow (o \oplus x \equiv x)\Big\rangle,$$
$$\Big\langle \text{"}\oplus\text{.inv.left"}, \ \forall_{\langle x\rangle} \underset{R}{\in}[x] \Rightarrow (x \oplus inv[x] \equiv o)\Big\rangle,$$
$$\Big\langle \text{"}\oplus\text{.commut"}, \ \forall_{\langle x, y\rangle} \Big(\underset{R}{\in}[x] \wedge \underset{R}{\in}[y]\Big) \Rightarrow ((x \oplus y) \equiv (y \oplus x))\Big\rangle,$$
$$\Big\langle \text{"}\otimes\text{.assoc"}, \ \forall_{\langle x, y, z\rangle} \Big(\underset{R}{\in}[x] \wedge \underset{R}{\in}[y] \wedge \underset{R}{\in}[z]\Big) \Rightarrow ((x \otimes y) \otimes z \equiv x \otimes (y \otimes z))\Big\rangle,$$
$$\Big\langle \text{"distrib.left"}, \ \forall_{\langle x, y, z\rangle} \Big(\underset{R}{\in}[x] \wedge \underset{R}{\in}[y] \wedge \underset{R}{\in}[z]\Big) \Rightarrow (x \oplus y) \otimes z \equiv (x \otimes z) \oplus (y \otimes z)\Big\rangle,$$
$$\Big\langle \text{"distrib.right"}, \ \forall_{\langle x, y, z\rangle} \Big(\underset{R}{\in}[x] \wedge \underset{R}{\in}[y] \wedge \underset{R}{\in}[z]\Big) \Rightarrow z \otimes (x \oplus y) \equiv (z \otimes x) \oplus (z \otimes y)\Big\rangle$$
$$\Big\rangle;$$

Having the Ring property defined we can assume that a given domain *A* is a Ring, as follows:

$$\text{Assume}\Big[\text{Ring}\Big[C, \text{"}\underset{C}{\oplus}\text{"}, \underset{C}{o}, \underset{C}{inv}, \text{"}\underset{C}{\otimes}\text{"}\Big]\Big];$$

4. Proving Domain Properties

For proving domain properties the Functor Natural Deduction Prover *(FunctorProver)* is provided. This prover is the outcome of our experiments with proving *knowledge transport* properties (see Section 1) and is currently under development. One aspect of the work presented in this section is detecting the elements of proof control which can be deduced from the domain structure of the mathematical objects we manipulate.

4.1 Calling the prover

Like most other provers in the *Theorema* system, *FunctorProver* is called with 2 arguments:

◇ The goal formula *G* to be proven
◇ The sequence *KB* of hypotheses (labeled formulae) used for proving *G,* referred to in the sequel as knowledge base.

For example in order to prove that the complexification *C* of a ring *A* is also a ring we can call the functor prover as follows:

$$\texttt{FunctorProver}\Big[\Big\langle \texttt{"F"},\ \Big(\texttt{Ring}\Big[\texttt{A},\ \texttt{"}\underset{A}{\oplus}\texttt{"},\ \underset{A}{\texttt{o}},\ \underset{A}{\texttt{inv}},\ \texttt{"}\underset{A}{\otimes}\texttt{"}\Big] \wedge (\texttt{C} \equiv \texttt{Complexification[A]})\Big) \Rightarrow \texttt{Ring}\Big[\texttt{C},\ \texttt{"}\underset{C}{\oplus}\texttt{"},\ \underset{C}{\texttt{o}},\ \underset{C}{\texttt{inv}},\ \texttt{"}\underset{C}{\otimes}\texttt{"}\Big]\Big\rangle,\ \langle\rangle\Big];$$

The knowledge base effectively used in proving the given formula is an extended one, consisting in:
◇ the formulae contained in the second argument to the prover (the explicit knowledge base),
◇ all domain properties declared in the current Mathematica session using the *Assume* function and
◇ all domain definitions in the current Mathematica session.
Hence we could write the call above in several steps as follows:

$$\texttt{Assume}\Big[\texttt{Ring}\Big[\texttt{A},\ \texttt{"}\underset{A}{\oplus}\texttt{"},\ \underset{A}{\texttt{o}},\ \underset{A}{\texttt{inv}},\ \texttt{"}\underset{A}{\otimes}\texttt{"}\Big]\Big];$$
$$\texttt{C} \sim\!\texttt{DefineDomain}\!\sim \texttt{Complexification[A];}$$
$$\texttt{FunctorProver}\Big[\Big\langle \texttt{"F"},\ \texttt{Ring}\Big[\texttt{C},\ \texttt{"}\underset{C}{\oplus}\texttt{"},\ \underset{C}{\texttt{o}},\ \underset{C}{\texttt{inv}},\ \texttt{"}\underset{C}{\otimes}\texttt{"}\Big]\Big\rangle,\ \langle\rangle\Big];$$

This is convenient in case of large knowledge bases, in order to keep the call to the functor prover small. The function *GetDomainProperties* can be used for checking the properties which are being assumed about a domain at any given time.
FunctorProver produces as output a notebook containing the proof or the proof attempt. The notebook is based on a nested−list proof object which has a format similar to the one described in [Buchberger, Jebelean 1997].
If called with the third argument "proof−object", *FunctorProver* will return the proof object obtained rather than producing a notebook output.
The output produced for the proof above is shown in Appendix A. Please note that due to lack of space we only show the proof of one ring axiom, the other subproofs are listed as closed cells. The italicized text has been added by hand to explain provers' interaction and control (see section 4.2). All other output was generated by *FunctorProver*.

4.2 Structure of the Prover

The largest part of the proof shown in Appendix A is produced by the predicate logic prover (see [Buchberger, Jebelean 1997]) and the simplification prover (see [Buchberger, Marin, Tomuta]). The functor prover performs "knowledge management" and control operations: it "evaluates" domain properties and definitions which appear in the knowledge base, i.e. it selects a subset of the logical formulae contained in the definition which is relevant to a particular prover, or in some cases it applies operator definitions.
The simplification prover is called only in terminal situations, in case the goal to be proven is brought to the form of an equality containing only constants or arbitrary but fixed variables. In such situations the functor prover passes the control to the simplification prover which attempts to prove the given subgoal and returns a proof object for the respective subproof. Hence, during the proving process the simplification prover appears as a "black box" for the functor prover.
The interaction with the natural deduction prover (further referred to as *PND*) is much more complex. Typically, in a proof natural deduction steps are mixed with steps in which definitions of properties, domains, domain operators are applied. To model this situation, a wrapper function called "*ProofStep*" was placed around each self−recursive call within the body of *PND*. In the default case (i.e. when *PND* is called as a standalone prover) this wrapper is defined as the identity function, thus resulting in a call to *PND.* In the functor module this wrapper is redefined to implement special handling of certain proof situations.
Such special handling consists in actions of the form:

◇ Analize current goal and knowledge base.
ì *If* some condition on the goal and knowledge base is fulfilled *then*
modify the goal and the knowledge base *and/or*
redefine *ProofStep and/or*
call another prover
◇ *otherwise* call PND with goal and knowledge base unchanged.

The proof objects corresponding to various subproofs produced by *PND* and simplification prover are embedded into the global proof object. Since the pretty−printer of the *Theorema* system is implemented as a set of rules for translating various proof situatios into English language, it can naturally handle such mixed proof objects. No further programming is necessary.

The analysis of the goal as well as the goal and knwoledge base modification represent control information which is largely domain dependant, hence reimplementing *ProofStep* means adapting various specialized provers to the domains we are working in. A topic for further study will be to implement *ProofStep* in a more clearly parametrised way with respect to the analysis and operations it performs such that it is easy to plug components into the main skeleton. This would enable us to do fast prototyping of domain−tailored provers and is a first step towards a high−level user−language for specifying domain−dependant proving techniques.
So far our focus was on proving knowledge transport properties, of the form shown in the introduction, where the goal is a universally quantified equality, but we have also considered pure natural deduction proofs which do not involve simplification, i.e. proving properties of relations in the newly defined domain.
In the sample proof given in the Appendix we explain, in italic text, the control elements implemented by the *ProofStep* function. All other text was automatically generated by the provers in our system.
The simplification prover orients equalities in the knowledge base with respect to a simplification ordering on ground terms. The ordering on ground terms is computed with respect to an ordering on the operators defined in a functor, from which an ordering for the operator symbols appearing in a given proof is computed. The equalities which are operator definitions are oriented from left to right so as to insure application of operator definitions in terms.

References

[Buchberger 1996a] B. Buchberger *Symbolic Computation: Computer Algebra and Logic,* in F.Baader and K.U., Schultz, Frontiers in Combining Systems, Kluwer Academic Publishers, 1996, p 193–220.

[Buchberger 1996b] B. Buchberger *An Induction Prover for Equalities over Natural Numbers,* Proceedings of the Theorema Workshop, June 9–10 1997, Hagenberg, Austria.

[Buchberger, Jebelean 1997] B. Buchberger, T.Jebelean *A Prover for Propositional Logic in Natural (Deduction) Style,* Proceedings of the Theorema Workshop, June 9–10 1997, Hagenberg, Austria.

[Buchberger, Kriftner 1997] B. Buchberger, F. Kriftner *Theorema: The Language.* Proceedings of the Theorema Workshop, June 9–10 1997, Hagenberg, Austria.

[Buchberger, Marin 1997] B. Buchberger, M. Marin, *Proving by Simplification,* Proceedings of the Theorema Workshop, June 9–10 1997, Hagenberg, Austria.

[Buchberger, Vasaru 1997] B. Buchberger, D.Vasaru, *An Inductive Prover for Equalities over Lists,* Proceedings of the Theorema Workshop, June 9–10 1997, Hagenberg, Austria.

Appendix A. Proof example

The natural deduction prover (PND) is called.

We prove (F).

We show

(F7)

$$\mathtt{Ring}\left[A, \underset{A}{\oplus}, \underset{A}{\circ}, \underset{A}{\mathtt{inv}}, \underset{A}{\otimes}\right] \wedge \mathtt{RingProperties}\left[A, \underset{A}{\oplus}, \underset{A}{\circ}, \underset{A}{\mathtt{inv}}, \underset{A}{\otimes}\right] \wedge (C \equiv \mathtt{Complexification}[A]) \Rightarrow \mathtt{Ring}\left[C, \underset{C}{\oplus}, \underset{C}{\circ}, \underset{C}{\mathtt{inv}}, \underset{C}{\otimes}\right]$$

under the assumptions

() $\langle\rangle$.

We prove (F) by the deduction rule.

We assume

(F7.H)

$$\mathtt{Ring}\left[A, \underset{A}{\oplus}, \underset{A}{\circ}, \underset{A}{\mathtt{inv}}, \underset{A}{\otimes}\right] \wedge \mathtt{RingProperties}\left[A, \underset{A}{\oplus}, \underset{A}{\circ}, \underset{A}{\mathtt{inv}}, \underset{A}{\otimes}\right] \wedge (C \equiv \mathtt{Complexification}[A])$$

and show

(F7.C) $\mathtt{Ring}\left[C, \underset{C}{\oplus}, \underset{C}{\circ}, \underset{C}{\mathtt{inv}}, \underset{C}{\otimes}\right]$.

The goal and knowledge base have been reduced to atomic property definitions and domain definitions. The target property definition (Ring) and the target domain definition (Complexification) are expanded. PND is called with the new goal and knowledge base.

We expand the domain definition and properties

$$\left\langle \mathtt{Ring}\left[C, \underset{C}{\oplus}, \underset{C}{\circ}, \underset{C}{\mathtt{inv}}, \underset{C}{\otimes}\right], \mathtt{Complexification}[A] \right\rangle$$

Thus we have to prove

(F7.C)

$$\left\langle \texttt{F7.C.C.}\oplus\texttt{.assoc}, \forall_{\langle x,y,z\rangle} \underset{C}{\in}[x] \wedge \underset{C}{\in}[y] \wedge \underset{C}{\in}[z] \Rightarrow \left(\left(x \underset{C}{\oplus} y\right) \underset{C}{\oplus} z\right) \equiv \left(x \underset{C}{\oplus} \left(y \underset{C}{\oplus} z\right)\right)\right\rangle \wedge$$
$$\left\langle \texttt{F7.C.C.}\oplus\texttt{.o.left}, \forall_{\langle x\rangle} \underset{C}{\in}[x] \Rightarrow \left(\underset{C}{o} \underset{C}{\oplus} x\right) \equiv x\right\rangle \wedge$$
$$\left\langle \texttt{F7.C.C.}\oplus\texttt{.inv.left}, \forall_{\langle x\rangle} \underset{C}{\in}[x] \Rightarrow \left(x \underset{C}{\oplus} \underset{C}{\mathrm{inv}}[x]\right) \equiv \underset{C}{o}\right\rangle \wedge$$
$$\left\langle \texttt{F7.C.C.}\oplus\texttt{.commut}, \forall_{\langle x,y\rangle} \underset{C}{\in}[x] \wedge \underset{C}{\in}[y] \Rightarrow \left(x \underset{C}{\oplus} y\right) \equiv \left(y \underset{C}{\oplus} x\right)\right\rangle \wedge$$
$$\left\langle \texttt{F7.C.C.}\otimes\texttt{.assoc}, \forall_{\langle x,y,z\rangle} \underset{C}{\in}[x] \wedge \underset{C}{\in}[y] \wedge \underset{C}{\in}[z] \Rightarrow \left(\left(x \underset{C}{\otimes} y\right) \underset{C}{\otimes} z\right) \equiv \left(x \underset{C}{\otimes} \left(y \underset{C}{\otimes} z\right)\right)\right\rangle \wedge$$
$$\left\langle \texttt{F7.C.C.distrib.left}, \forall_{\langle x,y,z\rangle} \underset{C}{\in}[x] \wedge \underset{C}{\in}[y] \wedge \underset{C}{\in}[z] \Rightarrow \left(\left(x \underset{C}{\oplus} y\right) \underset{C}{\otimes} z\right) \equiv \left(\left(x \underset{C}{\otimes} z\right) \underset{C}{\oplus} \left(y \underset{C}{\otimes} z\right)\right)\right\rangle \wedge$$
$$\left\langle \texttt{F7.C.C.distrib.right}, \forall_{\langle x,y,z\rangle} \underset{C}{\in}[x] \wedge \underset{C}{\in}[y] \wedge \underset{C}{\in}[z] \Rightarrow \left(z \underset{C}{\otimes} \left(x \underset{C}{\oplus} y\right)\right) \equiv \left(\left(z \underset{C}{\otimes} x\right) \underset{C}{\oplus} \left(z \underset{C}{\otimes} y\right)\right)\right\rangle$$

under the assumptions

$$\left\langle\langle\rangle, \left\langle \texttt{F7.H.1}, \mathrm{Ring}\left[A, \underset{A}{\oplus}, \underset{A}{o}, \underset{A}{\mathrm{inv}}, \underset{A}{\otimes}\right]\right\rangle, \left\langle \texttt{F7.H.2}, \mathrm{RingProperties}\left[A, \underset{A}{\oplus}, \underset{A}{o}, \underset{A}{\mathrm{inv}}, \underset{A}{\otimes}\right]\right\rangle,\right.$$
$$\left\langle \texttt{F7.H.3.Complexification.inv}, \forall_{\langle x,y\rangle} \underset{C}{\mathrm{inv}}[\langle x, y\rangle] \equiv \left\langle \underset{A}{\mathrm{inv}}[x], \underset{A}{\mathrm{inv}}[y]\right\rangle\right\rangle,$$
$$\left\langle \texttt{F7.H.3.Complexification.member}, \forall_{\langle x\rangle} \underset{C}{\in}[x] \Leftrightarrow \exists_{\langle xr,xi\rangle} \underset{A}{\in}[xr] \wedge \underset{A}{\in}[xi] \wedge (x \equiv \langle xr, xi\rangle)\right\rangle,$$
$$\left\langle \texttt{F7.H.3.Complexification.o}, \underset{C}{o} \equiv \left\langle \underset{A}{o}, \underset{A}{o}\right\rangle\right\rangle,$$
$$\left\langle \texttt{F7.H.3.Complexification.}\otimes, \forall_{\langle x1,y1,x2,y2\rangle} \left(\langle x1, y1\rangle \underset{C}{\otimes} \langle x2, y2\rangle\right) \equiv\right.$$
$$\left.\left\langle\left(\left(x1 \underset{A}{\otimes} x2\right) \underset{A}{\oplus} \underset{A}{\mathrm{inv}}\left[\left(y1 \underset{A}{\otimes} y2\right)\right]\right), \left(\left(x1 \underset{A}{\otimes} y2\right) \underset{A}{\oplus} \left(x2 \underset{A}{\otimes} y1\right)\right)\right\rangle\right\rangle,$$
$$\left.\left\langle \texttt{F7.H.3.Complexification.}\oplus, \forall_{\langle x1,y1,x2,y2\rangle} \left(\langle x1, y1\rangle \underset{C}{\oplus} \langle x2, y2\rangle\right) \equiv \left\langle\left(x1 \underset{A}{\oplus} x2\right), \left(y1 \underset{A}{\oplus} y2\right)\right\rangle\right\rangle\right\rangle$$

.

We split (F7.H.3.Complexification.member) into two implications.

We prove the conjunctive parts of(F7.C).

Proof of (F7.C.1)

$$\left\langle \texttt{F7.C.C.}\oplus\texttt{.assoc}, \forall_{\langle x,y,z\rangle} \underset{C}{\in}[x] \wedge \underset{C}{\in}[y] \wedge \underset{C}{\in}[z] \Rightarrow \left(\left(x \underset{C}{\oplus} y\right) \underset{C}{\oplus} z\right) \equiv \left(x \underset{C}{\oplus} \left(y \underset{C}{\oplus} z\right)\right)\right\rangle:$$

For proving (F7.C.1.F7.C.C.⊕.assoc) we prove, for arbitrary but fixed values,

(F7.C.1.F7.C.C.⊕.assoc’) $\underset{C}{\in}[x_0] \wedge \underset{C}{\in}[y_0] \wedge \underset{C}{\in}[z_0] \Rightarrow \left(\left(x_0 \underset{C}{\oplus} y_0\right) \underset{C}{\oplus} z_0\right) \equiv \left(x_0 \underset{C}{\oplus} \left(y_0 \underset{C}{\oplus} z_0\right)\right)$.

We prove (F7.C.1.F7.C.C.⊕.assoc’) by the deduction rule.

We assume

(F7.C.1.F7.C.C.⊕.assoc’.H) $\underset{C}{\in}[x_0] \wedge \underset{C}{\in}[y_0] \wedge \underset{C}{\in}[z_0]$

and show

(F7.C.1.F7.C.C.⊕.assoc’.C) $\left(\left(x_0 \underset{C}{\oplus} y_0\right) \underset{C}{\oplus} z_0\right) \equiv \left(x_0 \underset{C}{\oplus} \left(y_0 \underset{C}{\oplus} z_0\right)\right)$.

From (F7.H.3.Complexification.member.⇒) by specialization we obtain

(F7.H.3.Complexification.member.⇒/1) $\underset{C}{\in}[x_0] \Rightarrow \exists_{\langle xr,xi\rangle} \underset{A}{\in}[xr] \wedge \underset{A}{\in}[xi] \wedge (x_0 \equiv \langle xr, xi\rangle)$.

From (F7.C.1.F7.C.C.⊕.assoc'.H.1) and (F7.H.3.Complexification.member.⇒/1) we obtain by modus ponens

(F7.H.3.Complexification.member.⇒/1.C) $\exists_{\langle xr,xi\rangle} \underset{A}{\in}[xr] \wedge \underset{A}{\in}[xi] \wedge (x_0 \equiv \langle xr, xi\rangle)$.

By (F7.H.3.Complexification.member.⇒/1.C) we can take appropriate values such that

(F7.H.3.Complexification.member.⇒/1.C') $\underset{A}{\in}[xr_0] \wedge \underset{A}{\in}[xi_0] \wedge (x_0 \equiv \langle xr_0, xi_0\rangle)$.

From (F7.H.3.Complexification.member.⇒) by specialization we obtain

(F7.H.3.Complexification.member.⇒/1) $\underset{C}{\in}[y_0] \Rightarrow \exists_{\langle xr,xi\rangle} \underset{A}{\in}[xr] \wedge \underset{A}{\in}[xi] \wedge (y_0 \equiv \langle xr, xi\rangle)$.

From (F7.C.1.F7.C.C.⊕.assoc'.H.2) and (F7.H.3.Complexification.member.⇒/1) we obtain by modus ponens

(F7.H.3.Complexification.member.⇒/1.C) $\exists_{\langle xr,xi\rangle} \underset{A}{\in}[xr] \wedge \underset{A}{\in}[xi] \wedge (y_0 \equiv \langle xr, xi\rangle)$.

By (F7.H.3.Complexification.member.⇒/1.C) we can take appropriate values such that

(F7.H.3.Complexification.member.⇒/1.C') $\underset{A}{\in}[xr_1] \wedge \underset{A}{\in}[xi_1] \wedge (y_0 \equiv \langle xr_1, xi_1\rangle)$.

From (F7.H.3.Complexification.member.⇒) by specialization we obtain

(F7.H.3.Complexification.member.⇒/1) $\underset{C}{\in}[z_0] \Rightarrow \exists_{\langle xr,xi\rangle} \underset{A}{\in}[xr] \wedge \underset{A}{\in}[xi] \wedge (z_0 \equiv \langle xr, xi\rangle)$.

From (F7.C.1.F7.C.C.⊕.assoc'.H.3) and (F7.H.3.Complexification.member.⇒/1) we obtain by modus ponens

(F7.H.3.Complexification.member.⇒/1.C) $\exists_{\langle xr,xi\rangle} \underset{A}{\in}[xr] \wedge \underset{A}{\in}[xi] \wedge (z_0 \equiv \langle xr, xi\rangle)$.

By (F7.H.3.Complexification.member.⇒/1.C) we can take appropriate values such that

(F7.H.3.Complexification.member.⇒/1.C') $\underset{A}{\in}[xr_2] \wedge \underset{A}{\in}[xi_2] \wedge (z_0 \equiv \langle xr_2, xi_2\rangle)$.

The formula to be proven is an equality containing only constants and arbitrary but fixed variables. For all variables appearing in the goal, the membership function has been applied, giving their expression in terms of elements in the domain A. A new knowledge base is assembled and the simplification prover is called.

(F7.C.1.F7.C.C.⊕.assoc'.C) $\left(\left(x_0 \underset{C}{\oplus} y_0\right) \underset{C}{\oplus} z_0\right) \equiv \left(x_0 \underset{C}{\oplus} \left(y_0 \underset{C}{\oplus} z_0\right)\right)$

A proof by simplification works.

Simplification of the lhs term:

$\left(\left(x_0 \underset{C}{\oplus} y_0\right) \underset{C}{\oplus} z_0\right) =$ by (F7.H.3.Complexification.member.⇒/1.C'.3)

$\left(\left(\langle xr_0, xi_0\rangle \underset{C}{\oplus} y_0\right) \underset{C}{\oplus} z_0\right) =$ by (F7.H.3.Complexification.member.⇒/1.C'.3)

$\left(\left(\langle xr_0, xi_0\rangle \underset{C}{\oplus} \langle xr_1, xi_1\rangle\right) \underset{C}{\oplus} z_0\right) =$ by (Complexification.⊕)

$\left(\left\langle \left(xr_0 \underset{A}{\oplus} xr_1\right), \left(xi_0 \underset{A}{\oplus} xi_1\right)\right\rangle \underset{C}{\oplus} z_0\right) =$ by (F7.H.3.Complexification.member.⇒/1.C'.3)

$\left(\left\langle \left(xr_0 \underset{A}{\oplus} xr_1\right), \left(xi_0 \underset{A}{\oplus} xi_1\right)\right\rangle \underset{C}{\oplus} \langle xr_2, xi_2\rangle\right) =$ by (Complexification.⊕)

$\left\langle \left(\left(xr_0 \underset{A}{\oplus} xr_1\right) \underset{A}{\oplus} xr_2\right), \left(\left(xi_0 \underset{A}{\oplus} xi_1\right) \underset{A}{\oplus} xi_2\right)\right\rangle =$ by (A.⊕.assoc)

$\left\langle \left(\mathtt{xr}_0 \underset{A}{\oplus} \left(\mathtt{xr}_1 \underset{A}{\oplus} \mathtt{xr}_2\right)\right),\ \left(\left(\mathtt{xi}_0 \underset{A}{\oplus} \mathtt{xi}_1\right) \underset{A}{\oplus} \mathtt{xi}_2\right)\right\rangle =$ by (A.⊕.assoc)

$\left\langle \left(\mathtt{xr}_0 \underset{A}{\oplus} \left(\mathtt{xr}_1 \underset{A}{\oplus} \mathtt{xr}_2\right)\right),\ \left(\mathtt{xi}_0 \underset{A}{\oplus} \left(\mathtt{xi}_1 \underset{A}{\oplus} \mathtt{xi}_2\right)\right)\right\rangle \rfloor$

Simplification of the rhs term:

$\left(\mathtt{x}_0 \underset{C}{\oplus} \left(\mathtt{y}_0 \underset{C}{\oplus} \mathtt{z}_0\right)\right) =$ by (F7.H.3.Complexification.member.⇒/1.C'.3)

$\left(\langle \mathtt{xr}_0, \mathtt{xi}_0 \rangle \underset{C}{\oplus} \left(\mathtt{y}_0 \underset{C}{\oplus} \mathtt{z}_0\right)\right) =$ by (F7.H.3.Complexification.member.⇒/1.C'.3)

$\left(\langle \mathtt{xr}_0, \mathtt{xi}_0 \rangle \underset{C}{\oplus} \left(\langle \mathtt{xr}_1, \mathtt{xi}_1 \rangle \underset{C}{\oplus} \mathtt{z}_0\right)\right) =$ by (F7.H.3.Complexification.member.⇒/1.C'.3)

$\left(\langle \mathtt{xr}_0, \mathtt{xi}_0 \rangle \underset{C}{\oplus} \left(\langle \mathtt{xr}_1, \mathtt{xi}_1 \rangle \underset{C}{\oplus} \langle \mathtt{xr}_2, \mathtt{xi}_2 \rangle\right)\right) =$ by (Complexification.⊕)

$\left(\langle \mathtt{xr}_0, \mathtt{xi}_0 \rangle \underset{C}{\oplus} \left\langle \left(\mathtt{xr}_1 \underset{A}{\oplus} \mathtt{xr}_2\right),\ \left(\mathtt{xi}_1 \underset{A}{\oplus} \mathtt{xi}_2\right)\right\rangle\right) =$ by (Complexification.⊕)

$\left\langle \left(\mathtt{xr}_0 \underset{A}{\oplus} \left(\mathtt{xr}_1 \underset{A}{\oplus} \mathtt{xr}_2\right)\right),\ \left(\mathtt{xi}_0 \underset{A}{\oplus} \left(\mathtt{xi}_1 \underset{A}{\oplus} \mathtt{xi}_2\right)\right)\right\rangle \rfloor$

The proof object returned by the simplifier is embedded into the global one. The next proof of the next subgoal is started.

Proof of (F7.C.2) $\left\langle \mathtt{F7.C.C.\oplus.o.left},\ \forall_{\langle x\rangle} \underset{C}{\in}[\mathtt{x}] \Rightarrow \left(\underset{C}{\mathtt{o}} \underset{C}{\oplus} \mathtt{x}\right) \equiv \mathtt{x}\right\rangle$:

Proof of (F7.C.3) $\left\langle \mathtt{F7.C.C.\oplus.inv.left},\ \forall_{\langle x\rangle} \underset{C}{\in}[\mathtt{x}] \Rightarrow \left(\mathtt{x} \underset{C}{\oplus} \underset{C}{\mathtt{inv}}[\mathtt{x}]\right) \equiv \underset{C}{\mathtt{o}}\right\rangle$:

Proof of (F7.C.4) $\left\langle \mathtt{F7.C.C.\oplus.commut},\ \forall_{\langle x,y\rangle} \underset{C}{\in}[\mathtt{x}] \wedge \underset{C}{\in}[\mathtt{y}] \Rightarrow \left(\mathtt{x} \underset{C}{\oplus} \mathtt{y}\right) \equiv \left(\mathtt{y} \underset{C}{\oplus} \mathtt{x}\right)\right\rangle$:

Proof of (F7.C.5)

$\left\langle \mathtt{F7.C.C.\otimes.assoc},\ \forall_{\langle x,y,z\rangle} \underset{C}{\in}[\mathtt{x}] \wedge \underset{C}{\in}[\mathtt{y}] \wedge \underset{C}{\in}[\mathtt{z}] \Rightarrow \left(\left(\mathtt{x} \underset{C}{\otimes} \mathtt{y}\right) \underset{C}{\otimes} \mathtt{z}\right) \equiv \left(\mathtt{x} \underset{C}{\otimes} \left(\mathtt{y} \underset{C}{\otimes} \mathtt{z}\right)\right)\right\rangle$:

Proof of (F7.C.6)

$\left\langle \mathtt{F7.C.C.distrib.left},\ \forall_{\langle x,y,z\rangle} \underset{C}{\in}[\mathtt{x}] \wedge \underset{C}{\in}[\mathtt{y}] \wedge \underset{C}{\in}[\mathtt{z}] \Rightarrow \left(\left(\mathtt{x} \underset{C}{\oplus} \mathtt{y}\right) \underset{C}{\otimes} \mathtt{z}\right) \equiv \left(\left(\mathtt{x} \underset{C}{\otimes} \mathtt{z}\right) \underset{C}{\oplus} \left(\mathtt{y} \underset{C}{\otimes} \mathtt{z}\right)\right)\right\rangle$:

Proof of (F7.C.7)

$\left\langle \mathtt{F7.C.C.distrib.right},\ \forall_{\langle x,y,z\rangle} \underset{C}{\in}[\mathtt{x}] \wedge \underset{C}{\in}[\mathtt{y}] \wedge \underset{C}{\in}[\mathtt{z}] \Rightarrow \left(\mathtt{z} \underset{C}{\otimes} \left(\mathtt{x} \underset{C}{\oplus} \mathtt{y}\right)\right) \equiv \left(\left(\mathtt{z} \underset{C}{\otimes} \mathtt{x}\right) \underset{C}{\oplus} \left(\mathtt{z} \underset{C}{\otimes} \mathtt{y}\right)\right)\right\rangle$
: