Automated Higher-order Reasoning about Quantales

Han-Hing Dang, Peter Höfner
Institut für Informatik, Universität Augsburg
D-86135 Augsburg, Germany
{dang,hoefner}@informatik.uni-augsburg.de

Abstract

Originally developed as an algebraic characterisation for quantum mechanics, the algebraic structure of quantales nowadays finds widespread applications ranging from (non-commutative) logics to hybrid systems. We present an approach to bring reasoning about quantales into the realm of (fully) automated theorem proving. This will yield automation in various (new) fields of applications in the future. To achieve this goal and to receive a general approach (independent of any particular theorem prover), we use the TPTP Problem Library for higher-order logic. In particular, we give an encoding of quantales in the typed higher-order form (THF) and present some theorems about quantales which can be proved fully automatically. We further present prospective applications for our approach and discuss practical experiences using THF.

1 Introduction

Automated theorem proving (ATP) has brought automated reasoning into a wide variety of domains. Examples where significant and important success systems has been achieved with ATP, are software verification and mathematics. Full automation (without user interaction) is often only possible by first-order logic. For these tasks ATP systems like Vampire [26] and Prover9 [21] exist and SystemOnTPTP [28] provides a common language and a common interface. First-order ATP systems and the TPTP library have extensively been used by different users in various case studies covering various areas of sciences (e.g., [9, 15, 17, 18, 31]).

Recently, TPTP and SystemOnTPTP were extended to cover not only first-order reasoning but also reasoning within higher-order logic [6, 29]. A general aim is again full automation and no user interaction. As a part of TPTP, higher-order logic can now be expressed by the typed higher-order form (THF) which implements Church's simple theory of types [10]. This theory is based on the simply typed λ-calculus in which functional types are formed from basic types. The decision for simple type theory was made since this theory is already used as a common basis for a lot of higher-order ATP systems [6].

Up to now only a few case studies using THF exist. In this paper we provide a case study and bring reasoning about the algebraic structure of quantales into the realm of fully automated theorem proving. In particular, we encode quantales into the typed higher-order form of the TPTP library and perform a proof experiment using that approach with about 50 theorems. In doing so, we also evaluate all the ATP systems for higher-order reasoning that are integrated in SystemOnTPTP w.r.t. their capabilities for automated reasoning about quantales. In detail, these systems are IsabelleP [24], LEO-II [5], Satallax [2] and TPS [1]. We have chosen the structure of quantales due to several reasons:

- From a mathematical point of view quantales are special cases of the first-order structures of semirings and Kleene algebras. Since the latter structures are particular suitable for automated reasoning [15, 16, 17], the hope is that quantales are also suitable.

- Following [6], THF is particular suitable for set-based encodings. In this paper we will derive an encoding of quantales based on sets. Hence the hope is again that quantales yield good automation results.

- As a third reason why quantales are chosen, we mention the prospective applications. Originally, quantales were introduced to formalise phenomena of quantum mechanics. Later, this algebraic structure found various fields of application. Examples are classical logic like CTL and CTL* [22], non-classical logic like separation logic or non-commutative logic [12, 25, 32] as well as reasoning within hybrid systems [14]. Further examples can be found in [27].

There are two main contributions of the paper: First we develop the above-mentioned case study. As far as we know it is one of the largest proof experiments of THF. As a consequence, we show that fully automated reasoning within higher-order logic is feasible in principle. Unfortunately, none of the ATP systems used can even prove half of the given theorems. At the moment more complex properties cannot be verified with state-of-the-art ATP systems from the axioms. To achieve full automation in various new fields of applications, further development of fully automated ATP systems is needed. However, there was the same situation some years ago when TPTP offered the first problem suites for first-order logic. After TPTP was launched, the evolution of ATP systems were quite impressive, especially for the problems listed in the TPTP library. Since quantales will be part of TPTP v4.1.0 we hope for the same effect and believe that fully automated reasoning within quantales will be much more feasible in a couple of years. Due to this we will present, as the second contribution, a number of possible applications where automation in quantales can be applied if ATP systems perform better.

The paper is organised as follows: In Section 2 we define the algebraic structure of quantales and give all necessary mathematical background. In the following section we sketch the typed higher-order form of TPTP. We begin our case study by encoding quantales within THF in Section 4. After that we try to verify basic properties of quantales. In particular, we present and discuss the results of our proof experiment in Section 5. From this basic toolkit we then give possible fields of applications in Section 6. We conclude the paper by discussing some on-going and future work.

2 Quantales

Quantales, the algebraic basis for our case study, are used in a wide range of sciences like computer science, mathematics, physics or philosophy. Therefore they unify a wide range of applications from a mathematical point of view. More precisely, quantales are partially ordered sets that generalise various lattices of multiplicative ideals from ring theory as well as point free topologies and functional analysis. Later, in Section 6, we will discuss fields of application in much more detail.

Before defining quantales, we recapitulate some lattice theory.

A *complete lattice* $(S, \leq)$ is a partially ordered set in which all subsets have a supremum. The definition implies that all subsets have also an infimum and that there is a least element $0 =_{df} \bigsqcup \emptyset$ and a greatest element $\top =_{df} \bigsqcup S$. The infimum of an arbitrary set $X \subseteq S$ is denoted by $\bigsqcap X$ while the supremum is denoted by $\bigsqcup X$. The binary variants for two elements $x, y \in S$ are written as $x \sqcap y$ and $x \sqcup y$, resp.

A *quantale* (e.g. [27]) is a structure $(S, \leq, \cdot, 1)$ where $(S, \leq)$ is a complete lattice and $\cdot$ is an associative, completely disjunctive inner operation on S, i.e., $\cdot$ distributes over arbitrary suprema: For an index set I and arbitrary elements $x, y_i \in S$ we have

$$x \cdot (\bigsqcup_{i\in I} y_i) = \bigsqcup_{i\in I} (x \cdot y_i) \quad \text{and} \quad (\bigsqcup_{i\in I} y_i) \cdot x = \bigsqcup_{i\in I} (y_i \cdot x) \ . \tag{$*$}$$

Moreover 1 is required to be the identity of multiplication, i.e., $x \cdot 1 = 1 \cdot x = x$. The notion of a quantale is equivalent to Conway's notion of a *standard Kleene algebra* [11] and forms a special idempotent semiring. As an immediate consequence of the definition, multiplication is strict, i.e., $x \cdot 0 = 0 \cdot x = 0$ for all $x \in S$.

We now have a closer look at the axiomatisation of quantales. It is easy to see that the infinite distributivity laws $(*)$ make it nearly impossible to encode quantales within first-order logic. We are only aware of one possibility. However this would require many predicates and types and does not yield good results for automation. We follow the lines of Conway's book [11] and axiomatise quantales by the following set-based formulas.

$$\bigsqcup\emptyset = 0 \ , \quad (1) \qquad x \cdot 1 = 1 \cdot x = x \ , \quad (4)$$

$$\bigsqcup\{x\} = x \ , \quad (2) \qquad (x \cdot y) \cdot z = x \cdot (y \cdot z) \ , \quad (5)$$

$$\bigsqcup\{\bigsqcup X_i : i \in I\} = \bigsqcup\bigcup_{i\in I} X_i \ , \quad (3) \qquad \bigsqcup X_1 \cdot \bigsqcup X_2 = \bigsqcup\{x_1 \cdot x_2 : x_1 \in X_1, x_2 \in X_2\} \ , \quad (6)$$

where X_1, X_2, $X_i \subseteq S$ for all $i \in I \subseteq \mathbb{N}$, x, y, $z \in S$ and $\bigcup_{i\in I}$ denotes set union over an index set I. In this axiomatisation Axiom (1) characterises the special element 0. The Laws (2) and (3) "inductively" define the supremum; Equations (4) and (5) make the lattice to a multiplicative monoid. The last axiom is the counterpart for the infinite distributivity laws $(*)$. From these axioms one can easily define the order relation $\leq$ by $x \leq y \Leftrightarrow_{df} x \sqcup y = y$; the infimum operator can be characterised, as usual, by

$$\bigsqcap X = \bigsqcup\{y : \forall x \in X : y \leq x\} \ . \tag{7}$$

We will use the definitions of $\bigsqcup$, $\bigsqcap$ and $\cdot$ to encode quantales in the typed higher-order TPTP library in Section 4. Next to these operators Conway defines an operator * for finite iteration by $X^* = \{X^i : i \in \mathbb{N}\}$, where $X^0 = 1$ and $X^{n+1} = X^n \cdot X$. Since our approach follows Conway's axiomatisation, we would need arithmetic to encode this axiom. However THF does not allow arithmetic at the moment. Hence we do not discuss the star operator in this paper though we are aware of equivalent axioms that only need first-order logics (see [20]).

3 THF and Church's Simple Type Theory

In this section we sketch the higher-order approach in the TPTP problem library. We will only mention those points that are necessary for the encoding of quantales later on. In particular, we explain the meaning of the symbols that may occur in formulas. A detailed description of the higher-order approach can be found in [29].

The typed higher-order form (THF) of the TPTP problem library implements higher-order logic by Church's simple theory of types [10] since this theory is already used as a common basis for a lot of higher-order ATP systems [6]. That theory is based on the simply typed λ-calculus in which functional types are formed from basic types. These consist of types of *individuals* `$i`, of *Boolean values* `$o`; further ones can be built using the *function type constructor* `>`.

In THF all formulas are annotated and have the following form:

```
thf( <formula_name>, <role>, ( <formula> )).
```

`<formula_name>` identifies the formula by a unique name, the attribute `<role>` specifies the role of the formula like axiom, (type) definition, conjecture or theorem. The actual formula is given in the last part of the THF-structure. Detailed examples will be shown in the next section. Symbols that are allowed to occur in the formulas are `!`, `?` and `^`. They stand for $\forall$, $\exists$ and λ, resp. The binary operator `@` denotes function application.

4 Encoding in Higher-order TPTP-Syntax

As it can be seen from Section 2 in Conway's axiomatisation of quantales, a suitable encoding of set theory in higher-order logic is required. We have chosen an encoding that was already proposed by Benzmüller, Rabe and Sutcliffe [6]. This encoding has already been successfully implemented and applied. Sets are being represented by their characteristic functions. In particular, we use for an element x and a set X the following equivalence

$$x \in X \Leftrightarrow \mathsf{X}(x).$$

On the right-hand side `X` denotes a predicate. By this, set operations such as intersection or union can easily be expressed using the typed λ-calculus. For example, binary union is defined by

$$\lambda\, \mathsf{X}, \mathsf{Y}, x.\ ((\mathsf{X}\, x) \vee (\mathsf{Y}\, x))$$

assuming that the predicates `X` and `Y` have type $\alpha \rightarrow \texttt{\$o}$ and x has type α.

In the remainder we give an extract of the complete input file to demonstrate the encoding. A full encoding of the Axioms (1)–(6) is given in the Appendix[1] and at a web site [13]. We only consider the supremum definition here since it forms the most interesting part of the encoding.

```
14 thf(sup,type,(
15     sup: ( ( $i > $o ) > $i ) )).
```

The formula defines the type of the supremum operation. It takes a characteristic function of an arbitrary set as an argument (which has the type `( $i > $o )`) and returns its supremum of type `$i`. With this, the encoding of Axioms (1) and (2) is straightforward.

```
16 thf(sup_es,axiom,( (sup @ emptyset) = zero )).

17 thf(sup_singleset,axiom,(
18     ! [X: $i] : ( ( sup @ ( singleton @ X ) ) = X ) )).
```

Clearly, the function `emptyset` maps every element of type `$i` into false. Furthermore in the second formula `! [X: $i]` denotes a $\forall$-quantification over all elements `X` and `( singleton @ X )` a set containing only a single element `X`.

For the axiom $\bigsqcup\{\bigsqcup X_i : i \in I\} = \bigsqcup \bigcup_{i\in I} X_i$, we have to model characteristic functions representing the sets $\{\bigsqcup X_i : i \in I\}$ and $\bigcup_{i\in I} X_i$. This is done by defining functions that take a set of sets as an argument. Using the λ-calculus, the function for $\{\bigsqcup X_i : i \in I\}$ can be rephrased into

$$\lambda \mathsf{F}, x.\ \exists \mathsf{Y}.\ (\mathsf{F}\,\mathsf{Y}) \wedge \left(\left(\bigsqcup \mathsf{Y}\right) = x\right)$$

where $\mathsf{F} = \{X_i : i \in I\}$ and `Y` denotes a set X_j that contains x. This is directly encoded into THF.

[1]The line numbers we give in this section correspond to the one of the Appendix.

```
thf(supset,type,(
    supset: ( ( ( $i > $o ) > $o ) > $i > $o ) )).

thf(supset,definition,
    ( supset = ( ^ [F: ( $i > $o ) > $o, X: $i ] :
      ? [Y: $i > $o] : ( ( F @ Y ) & ( ( sup @ Y ) = X ) ) ) )).
```

In a similar way, a function `unionset` for $\bigcup_{i\in I} X_i$ can be given (cf. the Appendix). Using these functions, Axiom (3) can now encoded by

```
thf(sup_set,axiom,(
    ! [X: ( $i > $o ) > $o] : ( ( sup @ ( supset @ X ) ) =
                               ( sup @ ( unionset @ X ) ) ) )).
```

Arbitrary index sets I can now be handled by quantification over sets of sets. This is a big advantage of higher-order encodings. As mentioned before, there are attempts to encode quantales within first-order logic. However all known characterisations are very complex and very difficult to read. Set theory is encoded much more naturally in higher-order logic and therefore the axiomatisation for quantales we have given is quite natural. Moreover, it has been stated that set-theoretic theorems are solved more efficiently in higher-order logic than using first-order encodings [7].

5 Automating Basic Properties

In this section we use the encoding of Section 4 to automatically verify basic properties of quantales. We proved around 50 theorems, in particular theorems that involve infima and suprema over infinite sets. So far we have only proved a basic subset of the theorems we are interested in, since at the moment none of the ATP systems can even prove half of the them. Hence automating proofs of more complex properties in advanced fields of applications (cf. Section 6) are currently not possible. If not only the axioms but some further properties are provided, all listed theorems can be proved automatically.

For our experiment we used Sutcliffe's SystemOnTPTP Tool [28]. In particular we evaluated four higher-order logic ATP systems for finding proofs of theorems in quantales: IsabelleP 2009-1 [24], LEO-II 1.1 [5], Satallax 1.2 [2] and TPS 3.080227G1d [1]. The computers for the evaluation used a 2.8 GHz Intel Pentium 4 CPU, 1 GB of memory, running on a Linux 2.6 operating system. We set a CPU time limit of 300 s, which is known to be sufficient for the ATP systems to prove almost all the theorems they would be able to prove even with a significantly higher limit [30]. The results of this testing are shown in Table 1; a number indicates that a proof is found in that time, and a "–" indicates that the system reaches the time limit or gives up before reaching this limit.

In the remainder of the section we will discuss some of the results in more detail. In particular we report on some of the difficulties and practical aspects we faced when performing the proof experiment of Table 1 fully automatically.

Table 1 includes properties of $\bigsqcup$ and $\bigsqcap$ that denote the role of 0 being the least element w.r.t the natural order $\leq$ of the lattice structure. Moreover we encoded simple isotony properties ((E20)–(E25)), associativity (E18) and distributivity laws ((E35), (E37)–(E39)). At first we fed the ATP systems with simple theorems using the $\bigsqcup$ operation. We realised that in contrast to other ATP systems LEO-II timed out already when showing (E1) which could be immediately inferred by instantiating Axiom (2) setting $x = 0$. However since LEO-II is able to show

Property (E6) that can also be used together with (E5) to infer (E1) we think that either the given encoding is not appropriate for LEO-II or its search strategies are not effective enough for our tasks.

Looking at (E14) and (E15) that denote commutativity laws for $\bigsqcup$, Isabelle and TPS seem to have problems. The properties would be derivable from commutativity of set union which both systems could prove immediately.

Another difficulty arose when proving Theorem (E9) which simply denotes a binary variant of Axiom 3 ($\bigsqcup\{\bigsqcup X_i : i \in I\} = \bigsqcup \bigcup_{i\in I} X_i$). The axiom is quantified over sets of sets. None of the ATP systems was able to instantiate the axiom appropriately. To overcome this difficulty, an auxiliary function has been defined for building sets consisting of sets. By two additional assumptions, this function is related to ordinary set union (`unionset`).

	System	Isabelle	LEO-II	Satallax	TPS
(E1)	$\bigsqcup\{0\} = 0$	3.2	–	0.2	10.7
(E2)	$\bigsqcup(\{0\} \cup \{0\}) = 0$	3.1	–	92.1	–
(E3)	$\bigsqcup\{\bigsqcup\{x\}\} = x$	3.1	–	0.2	–
(E4)	$\bigsqcup(\{x\} \cup \emptyset) = x$	3.2	–	–	–
(E5)	$\bigsqcup\emptyset = \bigsqcup\{0\}$	3.1	–	0.2	18.3
(E6)	$\bigsqcup\emptyset = 0$	3.1	0.1	0.2	10.8
(E7)	$\bigsqcup(\{x\} \cup \{0\}) = \bigsqcup(\{\bigsqcup\{x\}\} \cup \{\bigsqcup\emptyset\})$	3.1	–	64.0	–
(E8)	$\bigsqcup(\{\bigsqcup\{x\}\} \cup \{\bigsqcup\{y\}\}) = \bigsqcup(\{x\} \cup \{y\})$	3.2	–	0.1	–
(E9)	$\bigsqcup(\{\bigsqcup X\} \cup \{\bigsqcup Y\}) = \bigsqcup(X \cup Y)$	–	–	–	–
(E10)	$\bigsqcup(\{\bigsqcup\{x\}\} \cup \{\bigsqcup\emptyset\}) = \bigsqcup(\{x\} \cup \emptyset)$	–	–	–	–
(E11)	$\bigsqcup(\{\bigsqcup\{x\}\} \cup \{\bigsqcup\emptyset\}) = x$	–	–	–	–
(E12)	$\bigsqcup(\{x\} \cup \{0\}) = x$	–	–	–	–
(E13)	$0 \le x$	–	–	–	–
(E14)	$\bigsqcup(\{x\} \cup \{y\}) = \bigsqcup(\{y\} \cup \{x\})$	–	0.1	0.8	–
(E15)	$x \sqcup y = y \sqcup x$	–	0.1	0.8	–
(E16)	$\bigsqcup(\{\bigsqcup\{x\}\} \cup (\{\bigsqcup\{y\}\} \cup \{\bigsqcup\{z\}\})) = \bigsqcup(\{x\} \cup (\{y\} \cup \{z\}))$	–	–	–	–
(E17)	$\bigsqcup((\{\bigsqcup\{x\}\} \cup \{\bigsqcup\{y\}\}) \cup \{\bigsqcup\{z\}\}) = \bigsqcup((\{x\} \cup \{y\}) \cup \{z\})$	–	–	–	–
(E18)	$(x \sqcup y) \sqcup z = x \sqcup (y \sqcup z)$	–	–	–	–
(E19)	$x \sqcup 0 = x$	–	–	–	–
(E20)	$x \le x \sqcup y$	–	–	–	–
(E21)	$x \in X \Rightarrow \bigsqcup X = \bigsqcup(X \cup \{x\})$	–	0.1	0.6	–
(E22)	$x \in X \Rightarrow \bigsqcup X = \bigsqcup(\{\bigsqcup X\} \cup \{x\})$	–	–	–	–
(E23)	$x \in X \Rightarrow \bigsqcup X = \bigsqcup X \sqcup x$	–	–	–	–
(E24)	$x \in X \Rightarrow x \le \bigsqcup X$	–	–	–	–
(E25)	$X \subseteq Y \Rightarrow \bigsqcup X \le \bigsqcup Y$	–	–	–	–
(E26)	$\{x \cdot y : x \in X, y \in \emptyset\} = \emptyset$	3.2	0.1	0.2	0.3
(E27)	$\{x \cdot y : x \in \emptyset, y \in Y\} = \emptyset$	3.3	0.1	0.2	0.3
(E28)	$\bigsqcup\{z\} \cdot \bigsqcup\emptyset = \bigsqcup\{x \cdot y : x \in \{z\}, y \in \emptyset\}$	3.3	–	0.3	18.8
(E29)	$0 \cdot \{z\} = \bigsqcup\{x \cdot y : x \in \emptyset, y \in \{z\}\}$	3.5	–	0.7	–
(E30)	$x \cdot 0 = 0$	–	–	–	–
(E31)	$0 \cdot x = 0$	–	–	–	–
(E32)	$\{x' \cdot y' : x' \in \{x\}, y' \in \{y, z\}\} = \{x \cdot y, x \cdot z\}$	3.5	0.1	–	0.3
(E33)	$\{x' \cdot y' : x' \in \{x, y\}, y' \in \{z\}\} = \{x \cdot z, y \cdot z\}$	3.4	0.1	–	0.3
(E34)	$x \cdot \bigsqcup(\{y\} \cup \{z\}) = \bigsqcup\{x' \cdot y' : x' \in \{x\}, y' \in \{y, z\}\}$	3.7	–	3.5	–
(E35)	$x \cdot (y \sqcup z) = x \cdot y \sqcup x \cdot z$	–	–	–	–
(E36)	$(\bigsqcup(\{x\} \cup \{y\})) \cdot z = \bigsqcup\{x' \cdot y' : x' \in \{x, y\}, y' \in \{z\}\}$	3.9	–	3.6	–
(E37)	$(x \sqcup y) \cdot z = x \cdot z \sqcup y \cdot z$	–	–	–	–
(E38)	$(x \sqcup y) \sqcap z = x \sqcap z \sqcup y \sqcap z$	–	–	–	–
(E39)	$x \sqcap (y \sqcup z) = x \sqcap y \sqcup x \sqcap z$	–	–	–	–
(E40)	$\bigsqcap\{0\} = 0$	–	–	–	–
(E41)	$\bigsqcap\emptyset = \top$	3.2	–	–	–
(E42)	$\bigsqcap\{x\} = x$	–	–	–	–
(E43)	$\bigsqcap(\{x\} \cap \emptyset) = \top$	3.2	–	–	–
(E44)	$\bigsqcap(\{\bigsqcap\{x\}\} \cup \{\bigsqcap\{y\}\}) = \bigsqcap(\{x\} \cup \{y\})$	–	–	–	–
(E45)	$\bigsqcap(\{\bigsqcap X\} \cup \{\bigsqcap Y\}) = \bigsqcap(X \cup Y)$	–	–	–	–
(E46)	$\bigsqcap(\{\bigsqcap\{x\}\} \cup \{\bigsqcap\emptyset\}) = x$	–	–	–	–
(E47)	$\bigsqcap(\bigsqcap\{x\} \sqcap \bigsqcap\emptyset) = x$	–	–	–	–
(E48)	$\top \cdot \top = \top$	–	–	–	–
(E49)	$x \le \top$	–	–	–	–
Proved		18	8	16	8

Table 1: Comparison of ATP systems for basic properties of quantals

```
thf(unionset_union,axiom,(
   ! [X: $i > $o, Y: $i > $o] : (
     ( unionset @ ( setofset @ X @ Y ) ) = ( union @ X @ Y ) ) )).
```

```
thf(sup_unionset_setofset,axiom,(
   ! [X: $i > $o, Y: $i > $o] : (
     ( sup @ ( unionset @ ( setofset @ X @ Y ) ) ) =
       ( sup @ ( unionset @ ( setofset @ ( singleton @ ( sup @ X ) ) @
       ( singleton @ ( sup @ Y ) ) ) ) ) ) )).
```

By this approach at least Isabelle was able to show this property.

For our experiment, we further encoded simple properties like (finite) associativity of $\bigsqcup$ (E18) or both annihilation laws ((E30), (E31)). None of the four ATP systems were able to show the theorems directly. This behaviour is a bit surprising since a proof by hand for (E30) simply uses Axioms (1), (2) and (6):

$$x \cdot 0 \;=\; \bigsqcup\{x\} \cdot \bigsqcup\emptyset \;=\; \bigsqcup\{x_1 \cdot x_2 : x_1 \in \{x\},\, x_2 \in \emptyset\} \;=\; \bigsqcup\emptyset \;=\; 0\ .$$

Therefore we extracted some steps of hand-written proofs and tried to show these separately. Using for example (E16) and (E17) as additional assumptions Isabelle is able to show (E18). With similar tricks we were able to prove all theorems of Table 1 fully automatically. This implies that one should add more properties than the pure axioms as assumptions.

These initial tasks with all the systems allow us to select the most powerful system for future applications, which are IsabelleP and Satallax at the moment. None of the ATP systems we included into our evaluation was even able to show half of the theorems we encoded. This could be due to an inappropriate encoding of our operations or due to inappropriate search strategies of the theorem provers used. For example the definition of `supset` gives the impression that the introduction of existentially quantified set variables `Y` leads to blind search and consequently bad results by increasing the state space.

6 Prospective Applications

Based on the given encoding we have proved a basic theorem kit for quantales. Together with the axioms the proved properties can be used as assumptions for automated theorem proving. In this section we sketch some of the prospective fields of applications where quantales are used and where our approach could be applied. Due to the complexity of these problems, tackling them seem not to be feasible with of-the-shelf theorem provers at the moment. However a further step in the evolution of fully automated higher-order ATP systems would enable us to perform these tasks.

Mathematics: Applications in mathematics are straightforward. As described in Section 2, quantales generalise various lattices of multiplicative ideals from ring theory as well as point free topologies and functional analysis. All these areas are possible places where THF can now be applied. Before tackling these problems one should start to verify more basic results on quantales and lattices as given in the foundational papers of Mulvey [23] and Rosenthal [27].

Logics: Quantales also occur in various logics. For computer scientist the branching time logic CTL* and its sub-logics CTL and LTL are the most prominent ones. In [22], a correspondence between quantales and these temporal logics is given. However, to realise reasoning in this setting arithmetic is needed since formulas like

$$\bigsqcup_{j\geq 0}\left(x^j \cdot y \sqcap \bigsqcap_{k<j} x^k \cdot z\right)$$

occur. We are not aware of any possibility of encoding properties like this without using arithmetics. For physicists, (non-commutatitive) linear logic is more suitable — its connection to quantales is discussed in [32]. Last but not least we want to mention that quantales are also used for reasoning about dynamic epistemic logic [3, 4]. This logic is used to model multi-agent systems and phenomena in philosophy.

Computer science: Besides reasoning in logic, quantales have further applications. For this paper we only mention hybrid systems — heterogeneous systems characterised by the interaction of discrete and continuous dynamics [14]. Algebraic reasoning with quantales can be used to verify properties about safety and liveness at an abstract level.

Physics: Originally, quantales were derived for modelling phenomena of quantum mechanics [8]. But quantales can also be used to formalise quantum logic — a logic defined for quantum physics [19, 25].

This closes our small list of prospective new applications for quantales where automated reasoning could be applied. However, as mentioned before, tackling these problems is not feasible at the moment since the performance of automated higher-order theorem provers is not yet sufficient.

A further application might be quantum computing since this is based on quantum mechanics. However, at the moment we are not aware of any formal treatment of quantum computing using quantales.

7 Conclusion and Outlook

We presented an approach to bring the algebraic structure of quantales into the realm of automated reasoning. This was done by using the higher-order approach of TPTP. In particular we presented an encoding in the typed higher-order form THF from which it was possible to prove a basic set of theorems about quantales. However, practical experience shows that at the moment only simple theorems can be proved; more complex properties need more assumptions as input or better search strategies for the ATP systems involved. We also presented prospective new applications for automated reasoning.

To perform the proof experiment, we used a set-based axiomatisation of quantales given by Conway. For future work, it would be interesting to investigate more suitable axiomatisations and more efficient encodings for the THF core since difficult theorems still need extra lemmas for full automation. Another research question is of course whether more efficient search strategies w.r.t. reasoning within quantales exist. To support the development of fully automated higher-order ATP systems, quantales will be part of the TPTP library v.4.1.0. This step hopefully helps to improve higher-order ATP systems for reasoning in algebraic structures similar to quantales within the near future.

Acknowledgements: We thank B. Möller, G. Sutcliffe and R. Glück for fruitful discussions and remarks.

References

[1] P. B. Andrews and C. E. Brown. TPS: A hybrid automatic-interactive system for developing proofs. *Journal of Applied Logic*, 4(4):367–395, 2006.

[2] J. Backes and C. Brown. Analytic tableaux for higher-order logic with choice. In J. Giesl and R. Haehnle, editors, *Automated Deduction — CADE-22*, LNAI, 2010. (to appear).

[3] A. Baltag, B. Coecke, and M. Sadrzadeh. Reasoning about dynamic epistemic logic. In W. van der Hoek, editor, *European Workshop on Multi-Agent Systems*, pages 605–614, 2004.

[4] A. Baltag, B. Coecke, and M. Sadrzadeh. Algebra and sequent calculus for epistemic actions. *ENTCS*, 126:27–52, 2005.

[5] C. Benzmüller, L. Paulson, F. Theiss, and A. Fietzke. The LEO-II project. In *Proceedings of the Fourteenth Workshop on Automated Reasoning, Bridging the Gap between Theory and Practice*, 2007.

[6] C. Benzmüller, F. Rabe, and G. Sutcliffe. THF0 — The Core of the TPTP Language for Higher-order Logic. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Automated Deduction*, number 5159 in LNAI, pages 491–506. Springer, 2008.

[7] C. Benzmüller, V. Sorge, M. Jamnik, and M. Kerber. Combined reasoning by automated cooperation. *Journal of Applied Logic*, 6(3):318–342, 2008.

[8] G. Birkhoff and J. von Neumann. The logic of quantum mechanics. *Annals of Mathematics*, 37:823–843, 1936. Reprint in [19].

[9] J. Bos. Applied Theorem Proving - Natural Language Testsuite. http://www.coli.uni-sb.de/ bos/atp/, 2000.

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

[11] J. H. Conway. *Regular Algebra and Finite Machines.* Chapman & Hall, 1971.

[12] H.-H. Dang, P. Höfner, and B. Möller. Towards algebraic separation logic. In R. Berghammer, A. Jaoua, and B. Möller, editors, *Relations and Kleene Algebra in Comp. Science*, volume 5827 of *LNCS*. Springer, 2009.

[13] P. Höfner. Database for automated proofs of Kleene algebra. `http://www.kleenealgebra.de` (accessed February 8, 2014).

[14] P. Höfner. *Algebraic Calculi for Hybrid Systems.* Books on Demand GmbH, 2009.

[15] P. Höfner and G. Struth. Automated reasoning in Kleene algebra. In F. Pfennig, editor, *Automated Deduction*, volume 4603 of *LNAI*, pages 279–294. Springer, 2007.

[16] P. Höfner and G. Struth. On automating the calculus of relations. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Automated Reasoning*, volume 5159 of *LNCS*, pages 50–66. Springer, 2008.

[17] P. Höfner, G. Struth, and G. Sutcliffe. Automated verification of refinement laws. *Annals of Mathematics and Artificial Intelligence, Special Issue on First-order Theorem Proving*, pages 35–62, 2008.

[18] A. Hommersom, P. Lucas, and P. van Bommel. Automated Theorem Proving for Quality-checking Medical Guidelines. In G. Sutcliffe, B. Fischer, and S. Schulz, editors, *Workshop on Empirically Successful Classical Automated Reasoning*, 2005.

[19] C. A. Hooker, editor. *The Logico-algebraic Approach to Quantum Mechanics.* D. Reidel Pub. Co., 1975.

[20] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110(2):366–390, 1994.

[21] W. W. McCune. Prover9 and Mace4. <`http://www.cs.unm.edu/~mccune/prover9`>. (accessed February 8, 2014).

[22] B. Möller, P. Höfner, and G. Struth. Quantales and temporal logics. In M. Johnson and V. Vene, editors, *Algebraic Methodology and Software Technology*, volume 4019 of *LNCS*, pages 263–277. Springer, 2006.

[23] C. Mulvey. &. *Rendiconti del Circolo Matematico di Palermo*, 12(2):99–104, 1986.

[24] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS.* Springer, 2002.

[25] J. Paseka and J. Rosicky. Quantales. In B. Coecke, D. Moore, and A. Wilce, editors, *Current Research in Operational Quantum Logic: Algebras, Categories and Languages*, volume 111 of *Fundamental Theories of Physics*, pages 245–262. Kluwer, 2000.

[26] A. Riazanov and A. Voronkov. The design and implementation of vampire. *AI Communications*, 15(2-3):91–110, 2002.

[27] K. Rosenthal. *Quantales and their Applications*, volume 234 of *Pitman Research Notes in Mathematics Series*. Longman Scientific & Technical, 1990.

[28] G. Sutcliffe. System description: SystemOnTPTP. In D. McAllester, editor, *Automated Deduction*, volume 1831 of *LNAI*, pages 406–410. Springer, 2000.

[29] G. Sutcliffe and C. Benzmüller. Automated reasoning in higher-order logic using the TPTP THF infrastructure. *Journal of Formalized Reasoning*, 2010. (to appear).

[30] G. Sutcliffe and C. Suttner. Evaluating general purpose automated theorem proving systems. *Artificial Intelligence*, 131(1-2):39–54, 2001.

[31] A. Wojcik. Formal Design Verification of Digital Systems. In *20th Design Automation Conference*, 1983.

[32] D. N. Yetter. Quantales and (noncommutative) linear logic. *Journal of Symbolic Logic*, 55(1):41–64, 1990.

A Complete THF-Encoding of Quantales

```
% --- Empty Set
    thf(emptyset_decl,type,(
      emptyset: $i > $o )).
    thf(emptyset,definition,
      ( emptyset  = ( ^ [X: $i] : $false ) )).
% --- Singleton Set
    thf(singleton_decl,type,(
      singleton: ( $i > $i > $o ) )).
    thf(singleton,definition,
      ( singleton  = ( ^ [X: $i,U: $i] : ( U = X ) ) )).
% --- Supremum
    thf(zero,type,(
      zero: $i )).
    thf(sup,type,(
      sup: ( ( $i > $o ) > $i ) )).
    thf(sup_es,axiom,(
      (sup @ emptyset) = zero )).
    thf(sup_singleset,axiom,(
      ! [X: $i] : ( ( sup @ ( singleton @ X ) ) = X ) )).
    thf(supset,type,(
      supset: ( ( ( $i > $o ) > $o ) > $i > $o ) )).
    thf(supset,definition,
      ( supset = ( ^ [F: ( $i > $o ) > $o, X: $i ] :
        ? [Y: $i > $o] : ( ( F @ Y ) & ( ( sup @ Y ) = X ) ) ) )).
    thf(unionset,type,(
      unionset: ( ( ( $i > $o ) > $o ) > $i > $o ) )).
    thf(unionset,definition,
      ( unionset  = ( ^ [F: ( $i > $o ) > $o, X: $i ] :
        ( ? [Y: $i > $o] : ( ( F @ Y ) & ( Y @ X ) ) ) ) )).
    thf(sup_set,axiom,(
       ! [X: ( $i > $o ) > $o] : ( ( sup @ ( supset @ X ) ) =
                                 ( sup @ ( unionset @ X ) ) ) )).
```

```
% --- Multiplication

    thf(multiplication,type,(
      multiplication: $i > $i > $i )).

    thf(crossmult,type,(
      crossmult: ( $i > $o ) > ( $i > $o ) > $i > $o )).

    thf(crossmult_def,definition,(
      crossmult = ( ^ [X: $i > $o,Y: $i > $o, A: $i] : (
        ? [X1: $i, Y1: $i] : ( ( X @ X1 ) & ( Y @ Y1 ) & ( A = ( multiplication @ X1 @ Y1 ) ) )
      ) ) )).

    thf(multiplication_sup,axiom,(
        ! [X: $i > $o, Y: $i > $o] : ( ( multiplication @ ( sup @ X ) @ ( sup @ Y ) )
        = ( sup @ ( crossmult @ X @ Y ) ) ) )).

    thf(one,type,(
      one: $i )).

    thf(multiplication_neutralr,axiom,(
        ! [X: $i] : ( ( multiplication @ X @ one ) = X ) )).

    thf(multiplication_neutrall,axiom,(
        ! [X: $i] : ( ( multiplication @ one @ X ) = X ) )).
```