Towards a Coq Formalization of a Quantified Modal Logic

Ana de Almeida Borges[1]

[1]Philosophy Department of the University of Barcelona, Montalegre 6. 08001 Barcelona, Catalonia, Spain

Abstract

We present a Coq formalization of the Quantified Reflection Calculus with one modality, or QRC_1. This is a decidable, strictly positive, and quantified modal logic previously studied for its applications in proof theory. The highlights are a deep embedding of QRC_1 in the Coq proof assistant, a mechanization of the notion of Kripke model with varying domains and a formalization of the soundness theorem. We focus on the design decisions inherent to the formalization and the insights that led to new and simplified proofs.

Keywords

Modal logic, strictly positive logic, Kripke semantics, feasible fragments, formalization, Coq

1. Introduction

The Quantified Reflection Calculus with one modality, denoted by QRC_1 and introduced in [1], is a strictly positive quantified modal logic inspired by the unimodal fragment of the Reflection Calculus, RC_1 [2, 3]. The quantified strictly positive language consists of a *verum* constant and relation symbols as atomic formulas, with the only available connectives being the conjunction, the diamond, and the universal quantifier. QRC_1 statements are assertions of the form $\varphi \rightsquigarrow \psi$ where φ and ψ are in this strictly positive language.

QRC_1 was born out of the wish for a nice quantified provability logic for theories of arithmetic such as Peano Arithmetic (PA), even though Vardanyan [4] showed that this is impossible in general. In fact, the full quantified provability logic of PA is Π^0_2-complete, and thus not recursively axiomatizable, let alone decidable. However, restricting the language to the strictly positive fragment is a viable solution [5].

The main results obtained for QRC_1 and described in [5] are soundness with respect to varying domain Kripke models, completeness for finite and constant domain Kripke models, and soundness and completeness with respect to two different (but related) arithmetical interpretations, marking it as a provability logic.

Here we report on an ongoing formalization [6] of part of the work presented in [5]. We will sometimes cite [1] as well, since it includes a more detailed, albeit less general, version of some of the same results. The current paper focuses on the formalization of the language

ARQNL 2022: Automated Reasoning in Quantified Non-Classical Logics, 11 August 2022, Haifa, Israel

ana.agvb@gmail.com (A. d. A. Borges)
https://aborges.eu (A. d. A. Borges)
0000-0001-5152-198X (A. d. A. Borges)

and axiomatization of QRC_1 (Sections 2 and 3 respectively), as well as of its Kripke semantics (Section 4) and soundness (Section 5). The formalization of the Kripke completeness is ongoing and will be described in a future work. The formalization of the arithmetical results has not been tackled yet.

1.1. Related Work

Quantified modal logic has been extensively studied [7], and even formalized. For example, [8] describes a modular Isabelle formalization of several quantified modal logics, including soundness and completeness theorems for them. On the other hand, [9] describes a set of Coq tactics to facilitate showing that a user-defined and possibly quantified modal logic proves a given statement. We have not made use of this library as our main goal was to prove meta-theorems of QRC_1, for which a deep embedding is more appropriate. There has also been work on a custom proof assistant for quantified modal logic [10], as well as an automated theorem prover for normal quantified modal logics [11].

Furthermore, there are several implementations of propositional modal logics, both in Coq [12, 13, 14] and in other proof assistants [15], as well as presentations of first-order logics [16, 17]. QRC_1 itself has never been mechanized before.

1.2. External Tools

Coq [18] is a general purpose interactive and formal proof management system. It provides a formal language expressive enough to write theorem statements and their proofs, as well as specifications of algorithms and their implementations. These proofs are verified by the Coq kernel, and are thus correct up to hypothetical (and unexpected) errors in the implementation of the kernel itself [19]. Coq has been extensively used to formalize both mathematical theorems [16, 20, 12, 21, 22, 23, 13] and software correctness [24, 25].

The core language is called Calculus of Inductive Constructions, a constructive type theory with support for inductive types, among other features. Even though the base theory is constructive, several common axioms are admissible, including excluded middle. We do not make use of any axioms in this development.

The Mathematical Components libraries, also known as MathComp [26], are libraries of formalized mathematics originally developed for the mechanization of the Four Color Theorem [20]. They serve as an alternative to Coq's standard library and provide the theories of basic types such as natural numbers and lists (`mathcomp-ssreflect`), as well as finite sets of so-called choice types (`mathcomp-finmap`, [27]). This development is based on MathComp and uses the SSReflect proof language [28].

Other interactive proof assistants could have been used to achieve similar results, but Coq provides many advantages. Its underlying theory is strong enough to prove our results, there are several well-developed libraries for many useful data structures, and the community is large and active. Furthermore, algorithms implemented in Coq can be extracted to other programming languages more suited for computation, such as OCaml. We do not make use of extraction in this development yet, but could do so in the future to obtain a certified and practical decision procedure for QRC_1.

1.3. Formalization

This paper tries to be accessible to someone who has never used Coq, or even other interactive proof assistants. For this reason, we mostly highlight the interesting design decisions and difficulties that would plausibly arise in other formalization efforts and stick to standard mathematical notation. The only exception is Section 4.1, where we briefly comment on a well-known issue with type hierarchies and the solution we implemented.

When possible, we mention the Coq name for each definition and theorem presented here. These names are hyperlinks to an online rendition of their source code. There is also a summary of the formalization available online,[1] serving as a kind of documentation.

In Coq, every term has a type, and every type is also a term (and thus has a (larger) type itself). There is a special type, called `Prop`, which is meant to represent logical propositions. Thus, when P : `Prop` we think of P as the statement of a lemma, and of inhabitants of P as proofs of P. Most of the time, we don't care which particular proof of P was used to show P was inhabited (i.e., proved).[2] In contrast, when defining a non-`Prop` object, we often do care about which specific inhabitant was chosen. For example, the statement 0 : `nat` is much more informative than the statement "`nat` is inhabited". We refer to inhabitants of `Prop` as proofs or non-informative terms, and to other objects as informative terms.

Even though it is possible, there are some issues with including proofs in the middle of otherwise-informative terms. It has been our experience that a Coq development becomes much simpler when this is avoided and informative terms are clearly separated from non-informative ones.[3] We only mix these when defining objects meant exclusively for theorem statements (as in Section 4.1), or when we couldn't find an alternative. Even then, postponing this mix as much as possible led to a clear improvement in the complexity of the implementation, as described in Section 5.2.

We briefly present some figures comparing this formalization with mathematical text describing the same definitions, theorems, and proofs. With this information, we calculate this project's de Bruijn factor [32], which is the quotient between the compressed size of the formalization and the compressed size of natural language text describing the same results. The formalization described in this document takes up about 10.8K of memory when compressed (corresponding to about 1200 lines of code), roughly 1.4 times as much as the compressed size of the LaTeX source for [1] (corresponding to about 8 pages). Strikingly, the ongoing formalization of the completeness theorem is already at 39.0K or 3800 lines of code (including the code shared with the soundness formalization), while the relevant LaTeX source is about 2.8 times smaller (14 pages long).

[1] https://ana-borges.gitlab.io/QRC1-Coq/v0.1.0/Summary.html

[2] The proof mining field [29] is a clear exception, although if one were to implement proof mining techniques in Coq, one would probably use something other than `Prop` to represent logical propositions.

[3] This is arguable and boils down to style. There is a well-known Coq textbook [30] describing the opposite strategy. Here we tried to follow the MathComp guidelines [31] instead.

2. Quantified and Strictly Positive Formulas

We define the names of variables, `VarName`, as simply the natural numbers, ensuring that we have a countable number of variables available. We then define the concept of `signature` as including a finite set of constant names, a finite set of predicate names, and a function from the predicate names to the natural numbers assigning an arity to each one. Our language includes no non-constant function symbols.

A `term` is either a variable or a constant. We define the appropriate canonical instances for `eqType` (equality on terms is decidable), `countType` (there is a countable amount of terms) and `choiceType` (there is a choice operator for terms). This makes it possible to talk about finite sets of terms using the machinery of the Finite Maps Library [27] later on.

A `formula` is either $\top$, a predicate name together with a tuple of terms of the arity given by the signature, a conjunction of two other formulas, a diamond of one other formula, or a universal quantifier of a variable and another formula. We use the standard mathematical notation in this text, and reasonable approximations for this notation in the Coq development.

The `Language.v` file then goes on to define several standard notions and facts about them, such as free variables (fv(φ) or `fv`), substitution ($\varphi[t_1 \leftarrow t_2]$ or `sub`), and being free for a variable in a formula (no occurrence of a free variable becomes bound after the substitution, or `freefor`), which we discuss in the next subsection.

2.1. Binders

Our formulas live in a quantified language, and as such there is a distinction between free and bound variables. This distinction is important when dealing with substitution, since it should not impact bound variables. Thus, $(\forall x\, \varphi)[x \leftarrow y]$ should be exactly $\forall x\, \varphi$ because x is not a free variable of that formula.

There is one tricky issue, though: cases where replacing a free variable by a term lead to a previously free occurrence becoming bound, such as in $(\forall y\, S(x, y))[x \leftarrow y]$. Here a naive substitution would lead to $\forall y\, S(y, y)$, which clearly does not preserve logical strength. In informal mathematics it is common to ignore this issue by observing that the names of the bound variables are ultimately irrelevant: if we wish to replace x by y in $\forall y\, S(x, y)$, then this can be achieved by first renaming the bound variable to some fresh name such as z, and then doing the substitution. The final formula would then be $\forall z\, S(y, z)$. This is the approach taken by O'Connor in his formalization of the Gödel-Rosser incompleteness theorem [16]. However, the paper cites this decision as having led to many issues in the formalization; although it ultimately works, we did not wish to use the same strategy.

Another common solution for this problem is to use de Bruijn indexes [33]. This avoids naming bound variables altogether, so this concern does not appear. However, this approach is complex in its own right and would make the formalization considerably different from the original paper.

There is a tool named Autosubst [34] that internally uses de Bruijn indexes but generates the boilerplate code by itself and thus cuts back on the complexity and size of the developments. We have not yet made use of Autosubst, but it would be interesting to see how many lines of code and complications it would save. We leave this as future work.

The approach we settled on was inspired by [35] and the will to avoid mixing non-informative and informative objects as explained in Section 1.3. We define unguarded substitution, `sub`, and add an extra assumption, `freefor`, as needed. This assumption assures us that if the substitution goes through then the replacing term will not be captured under any binders.

We already spoke of terms being free for variables in formulas in our previous work [1, 5], so the formalization is very similar to the informal mathematics. Furthermore, there were no significant complications in using this approach in the formalization of the Kripke soundness theorem for QRC_1. This was no longer the case for the formalization of the Kripke completeness theorem, but we postpone discussing this to a future work, when the formalization is completed.

One downside of this strategy is that variable names must be picked with some foresight. Going back to our example from above, if $(\forall y\, S(x,y))[x{\leftarrow}y]$ ever appears in our development then we can perform the substitution, but won't be able to use any of the results about it because here y is not free for x in $\forall y\, S(x,y)$. Thus it is assumed that in practice the names for the bound variables do not clash with the names for the free variables, or that bound variables are renamed as needed.

3. QRC_1

The axioms and rules of QRC_1 are defined in a deeply embedded way in the `QRC1.v` file, which also includes some proofs of simple QRC_1 facts.

Definition 3.1 (`QRC1Proof`)**.** Let φ, ψ, and χ be any quantified strictly positive formulas. The axioms and rules of QRC_1 are the following:

(i) $\varphi \rightsquigarrow \top$ and $\varphi \rightsquigarrow \varphi$;

(ii) $\varphi \wedge \psi \rightsquigarrow \varphi$ and $\varphi \wedge \psi \rightsquigarrow \psi$;

(iii) if $\varphi \rightsquigarrow \psi$ and $\varphi \rightsquigarrow \chi$, then $\varphi \rightsquigarrow \psi \wedge \chi$;

(iv) if $\varphi \rightsquigarrow \psi$ and $\psi \rightsquigarrow \chi$, then $\varphi \rightsquigarrow \chi$;

(v) if $\varphi \rightsquigarrow \psi$, then $\Diamond\varphi \rightsquigarrow \Diamond\psi$;

(vi) $\Diamond\Diamond\varphi \rightsquigarrow \Diamond\varphi$;

(vii) if $\varphi \rightsquigarrow \psi$, then $\varphi \rightsquigarrow \forall x\, \psi$ ($x \notin \mathrm{fv}(\varphi)$);

(viii) if $\varphi[x{\leftarrow}t] \rightsquigarrow \psi$, then $\forall x\, \varphi \rightsquigarrow \psi$ (t free for x in φ);

(ix) if $\varphi \rightsquigarrow \psi$, then $\varphi[x{\leftarrow}t] \rightsquigarrow \psi[x{\leftarrow}t]$ (t free for x in φ and ψ);

(x) if $\varphi[x{\leftarrow}c] \rightsquigarrow \psi[x{\leftarrow}c]$, then $\varphi \rightsquigarrow \psi$ (c not in φ nor ψ).

If $\varphi \rightsquigarrow \psi$, we say that ψ follows from φ in QRC_1.

We briefly comment on the above axioms and rules. The first six statements correspond to axioms and rules of RC_1, while the two quantifier rules are standard in first-order logic. The final two rules, called term instantiation and constant elimination respectively, fulfill an essential role in the completeness of QRC_1. The best way to think of them is as quantifier rules in disguise. Since our semantics (described in Section 4) interprets the free variables of both sides of $\rightsquigarrow$ in the same way, we can also think of such free variables as being generalized outside this implication. In other words, $P(x) \rightsquigarrow Q(x)$ can be thought of as $\forall x\, (P(x) \rightsquigarrow Q(x))$. We never explicitly

write the latter, since it falls outside the scope of the strictly positive language. However, we do wish to arrive at the conclusions such a formula promises, namely, we wish to be able to simultaneously instantiate x on both sides of $\rightsquigarrow$ by any term (accomplished by Rule 3.1.(ix)) and to simultaneously "generalize" a given term as well (accomplished by Rule 3.1.(x)).

We used a deep embedding to represent the axioms and rules of QRC_1, which facilitates the proofs of meta-theorems such as soundness and completeness. However, it is still quite easy to use this embedding to prove theorems of QRC_1 itself, as the simple formalization of the following lemma illustrates.

Lemma 3.2. The following are theorems (or derivable rules) of QRC_1:

(i) `AllC`: $\forall x\, \forall y\, \varphi \rightsquigarrow \forall y\, \forall x\, \varphi$;

(ii) `All_sub`: $\forall x\, \varphi \rightsquigarrow \varphi[x{\leftarrow}t]$ (t free for x in φ);

(iii) `Diam_All`: $\Diamond \forall x\, \varphi \rightsquigarrow \forall x\, \Diamond \varphi$;

(iv) `alphaconversion`: $\forall x\, \varphi \rightsquigarrow \forall y\, \varphi[x{\leftarrow}y]$ (y free for x in φ and $y \notin \mathrm{fv}(\varphi)$);

(v) `TermIr`: if $\varphi \rightsquigarrow \psi$, then $\varphi \rightsquigarrow \psi[x{\leftarrow}t]$ (x not free in φ and t free for x in ψ);

(vi) `Const_AllIr`: if $\varphi \rightsquigarrow \psi[x{\leftarrow}c]$, then $\varphi \rightsquigarrow \forall x\, \psi$ (x not free in φ and c not in φ nor ψ).

Like other provability logics, QRC_1 is irreflexive, i.e., $\varphi \rightsquigarrow \Diamond \varphi$ is not provable. However, unlike other provability logics, this fact can be proved without semantics. Its formalization is called `Diam_irreflexive`.

4. Kripke Semantics

The Kripke semantics for QRC_1 generalizes the Kripke semantics for propositional modal logics by transforming each world into a first-order model. Each of the worlds has its own domain, and the only restriction on the domains is that there must be a function between each pair fulfilling certain properties (cf. Definition 4.2). We present here the version implemented in Coq and comment on the slight discrepancies with the definition from [5] afterward.

Definition 4.1 (`rawFrame`, `rawModel`)**.** A *Kripke model* $\mathcal{M}$ in a signature Σ is a tuple $\langle W, R, \{M_w\}_{w \in W}, \{\eta_{w,u}\}_{w,u \in W}, \{I_w\}_{w \in W}, \{J_w\}_{w \in W} \rangle$ where:

- W is a finite set (the set of worlds, where individual worlds are referred to as w, u, v, etc);
- R is a binary relation on W (the accessibility relation);
- M_w is a finite set for each $w \in W$ (the domain of the world w, whose elements are referred to as d, d_0, d_1, etc);
- $\eta_{w,u}$ is a function from M_w to M_u for each $w, u \in W$ (the compatibility function between w and u);

- for each $w \in W$, the interpretation I_w assigns an element of the domain M_w to each constant $c \in \Sigma$, written c^{I_w}; and
- for each $w \in W$, the interpretation J_w assigns a set of n-tuples $S^{J_w} \subseteq \wp((M_w)^n)$ to each n-ary relation symbol $S \in \Sigma$.

The $\langle W, R, \{M_w\}_{w \in W}, \{\eta_{w,u}\}_{w,u \in W} \rangle$ part of the model is called its *frame*. We say that the frame (or model) is *constant domain* if all the M_w coincide and all the $\eta_{w,u}$ are the identity function.

The above definition of frame, called `rawFrame` because it is not necessarily adequate, is not exactly like the one presented in [5]. Note how above we postulate $\eta_{w,u}$ functions for every pair of worlds. In our previous work, $\eta_{w,u}$ was only defined when wRu. This made sense because the notion of satisfaction only uses the compatibility functions in those cases. However, including such a non-informative restriction in a Coq definition, although possible, leads to noticeable inconveniences, as described in Section 1.3.

Our work-around was to change the notion of frame so that functions $\eta_{w,u}$ must exist for every pair of worlds w and u. The fact that in principle we only make use of the ones between pairs of worlds connected through R is immaterial. We must add an extra assumption to the notion of adequate frame to maintain soundness, though: $\eta_{w,w}$ must be the identity for any world w.[4] This decision was crucial in the mechanization of the soundness of Rule 3.1.(x), which was the trickiest one. See Section 5.2 for more details.

Note how we do not lose generality with this alternative definition. The extra $\eta_{w,u}$ functions can obviously be dropped to obtain the original definition; on the other hand, as long as the domain M_u is non-empty,[5] we can define a function $\eta_{w,u}$ from M_w to M_u. Since this function will not be used, it doesn't matter which one we pick.

The other difference is that we only implement finite models, in the sense that both the set of worlds and each domain are finite. This does not impact the completeness proof, since QRC_1 has the finite model property [1], but it does mean that the formalized soundness proof is slightly weaker than the more general one presented in [5].

The relevant frames and models will need to satisfy a number of requisites.

Definition 4.2 (`adequateF`, `adequateM`)**.** A frame $\mathcal{F}$ is *adequate* if:

- R is transitive: if wRu and uRv, then wRv;
- the η functions respect transitivity: if wRu and uRv, then $\eta_{w,v}(d) = \eta_{u,v}(\eta_{w,u}(d))$ for every d in the domain of w; and
- the $\eta_{w,w}$ functions are the identity.

A model is *adequate* if it is based on an adequate frame and it is:

[4] This restriction was already implicit for any reflexive world w as a consequence of $\eta_{w,w}$ respecting transitivity in that case (see Definition 4.2).

[5] We never explicitly require non-empty domains, but a w-assignment g can only exist if the domain of w is non-empty. Thus, the soundness theorem holds vacuously for empty-domain models, and ignoring such models is not a loss.

- concordant: if wRu, then $c^{I_u} = \eta_{w,u}(c^{I_w})$ for every constant c.

Note that in an adequate and rooted model the interpretation of the constants is fully determined by their interpretation at the root.

The notion of `frame` is defined by pairing a `rawFrame` with a proof that it is `adequateF`, and similarly for `model`. We go into more technical details in Section 4.1.

We use assignments to define truth at a world in a first-order model. Fixing a world w, a w-`assignment` g is a function assigning a member of the domain M_w to each variable in the language.

Two w-assignments g and h are Γ-*alternative*, written $g \sim_\Gamma h$ (or `Xaltern g h Γ` in Coq), if they coincide on all variables other than the ones in Γ. We write $g \sim_x h$ instead of $g \sim_{\{x\}} h$. A w-assignment g is extended to terms by defining $g(c) := c^{I_w}$ for any constant c.

We now define satisfaction at a world.

Definition 4.3 (`sat`)**.** Let $\mathcal{M} = \langle W, R, \{M_w\}_{w\in W}, \{\eta_{w,u}\}_{w,u\in W}, \{I_w\}_{w\in W}, \{J_w\}_{w\in W}\rangle$ be a model in some signature Σ, and let $w \in W$ be a world, g be a w-assignment, S be an n-ary relation symbol, and φ, ψ be formulas in the language of Σ.

We define $\mathcal{M}, w \Vdash^g \varphi$ (φ is true at w under g) by induction on φ as follows.

- $\mathcal{M}, w \Vdash^g \top$;
- $\mathcal{M}, w \Vdash^g S(t_0, \ldots, t_{n-1})$ iff $\langle g(t_0), \ldots, g(t_{n-1})\rangle \in S^{J_w}$;
- $\mathcal{M}, w \Vdash^g \varphi \wedge \psi$ iff both $\mathcal{M}, w \Vdash^g \varphi$ and $\mathcal{M}, w \Vdash^g \psi$;
- $\mathcal{M}, w \Vdash^g \Diamond\varphi$ iff there is a $u \in W$ such that wRu and $\mathcal{M}, u \Vdash^{\eta_{w,u}\circ g} \varphi$;
- $\mathcal{M}, w \Vdash^g \forall x\, \varphi$ iff for all w-assignments h such that $h \sim_x g$, we have $\mathcal{M}, w \Vdash^h \varphi$.

Note how we haven't required that $\mathcal{M}$ be adequate in the definition of satisfaction, as it is not needed. We will of course assume the models are adequate when proving facts about them. Note also that the expression $\mathcal{M}, w \Vdash^g \varphi$ is only defined when g is a w-assignment.

The main results on QRC_1 are as follows.

Theorem 4.4 (`soundness`)**.** If $\varphi \rightsquigarrow \psi$, then for any adequate model $\mathcal{M}$, for any world $w \in W$, and for any w-assignment g:

$$\mathcal{M}, w \Vdash^g \varphi \implies \mathcal{M}, w \Vdash^g \psi.$$

Theorem 4.5 (Completeness, [5])**.** If $\varphi \not\rightsquigarrow \psi$, then there is an adequate, finite, constant domain and irreflexive model $\mathcal{M}$, a world $w \in W$, and a w-assignment g such that:

$$\mathcal{M}, w \Vdash^g \varphi \quad \text{and} \quad \mathcal{M}, w \nVdash^g \psi.$$

Since we have the finite model property, we can conclude that QRC_1 is decidable by Post's Theorem.

We focus on the (constructive and axiom-free) formalization of the soundness theorem in Section 5 and leave the formalization of the completeness theorem to a future work.

4.1. Type Hierarchies

When defining specific frames or models or operations on arbitrary frames or models such as Definition 4.3, we use the `raw` versions. On the other hand, when stating facts about frames or models we use the adequate versions, if necessary. We make use of implicit coercions in order to smoothly refer to operations that expect, for example, a `rawFrame` in a theorem statement about a `frame`.

A coercion is a function $f : A \to B$ that is automatically used by Coq when an otherwise ill-typed statement would be well-typed in the presence of f. For example, we declare a coercion from `rawFrame` to `world` (the set of worlds) that lets us write statements such as `forall (F : rawFrame), forall (w : F), ...` that closely resemble the common shorthand of stating that a world is part of a frame instead of part of the set of worlds of the frame. In this case Coq automatically infers the implicit coercion `world` necessary to make the statement type-check. Explicitly, it would be `forall (F : rawFrame), forall (w : world F), ...`

We use a small number of coercions in our development, the most important of which are represented in Figure 1. These coercions serve as a translation between a type and a super-type (in the sense that the former is a sub-type of the latter). We have a very small type hierarchy. Formalizations of, say, mathematical algebra or large libraries such as MathComp include rich hierarchies [36], and there are existing tools to implement and maintain such large hierarchies such as the Hierarchy Builder [37].

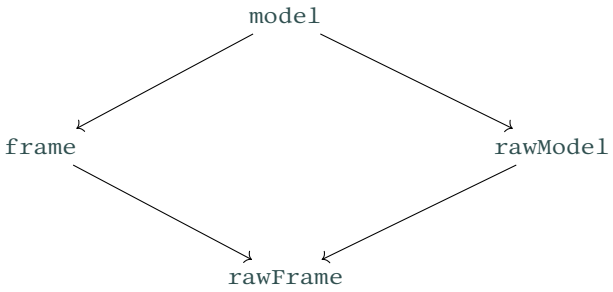

Figure 1: A representation of the four datatypes defined to represent frames and models and the coercions between them. Each arrow from `X` to `Y` represents the coercion `Y_of_X`.

Still, even with a small hierarchy we do run into some issues. For example, consider the following unification problem:

$$\texttt{rawFrame\_of\_frame}\ ?F = \texttt{rawFrame\_of\_rawModel}\ (\texttt{rawModel\_of\_model}\ M) \quad (1)$$

In words, given a `model` M, a `frame` $?F$ must be found such that its `rawFrame` corresponds to the `rawFrame` of the model M. The diamond represented in Figure 1 commutes, and so we define the canonical coercion `frame_of_model` as the path to solve (1) with $?F =$ `frame_of_model` M.

5. Soundness

We show the soundness of QRC_1 (Theorem 4.4) by induction on the proof of $\varphi \rightsquigarrow \psi$. Some of the axioms and rules are trivial, and we do not comment on them.

The soundness of the transitivity axiom (Axiom 3.1.(vi), `Trans`) follows from both the transitivity of R and the fact that the compatibility functions respect transitivity. We also use the fact that assignments only matter for variables appearing free in the formula (Lemma 5.1, further discussed in Section 5.1) to take advantage of the extensional equality of $\eta_{w,v}$ and $\eta_{u,v} \circ \eta_{w,u}$.

Lemma 5.1 (`sat_Xalternfv`)**.** Let $\mathcal{M}$ be an adequate model, w be any world, g, h be any Γ-alternative w-assignments, and φ be a formula with no free variables in Γ. Then:

$$\mathcal{M}, w \Vdash^{g} \varphi \iff \mathcal{M}, w \Vdash^{h} \varphi.$$

This lemma is all that is needed to show the soundness of the $\forall$-introduction on the right rule (Rule 3.1.(vii), `AllIr`). For both $\forall$-introduction on the left (Rule 3.1.(viii), `AllIl`) and term instantiation (Rule 3.1.(ix), `TermI`), we use the fact that a formula φ is valid under an assignment $\tilde{g}$ if and only if $\varphi[x \leftarrow t]$ is valid under an assignment g when $g \sim_x \tilde{g}$ and $\tilde{g}(x) = g(t)$ (Lemma 5.2) as the main building block.

Lemma 5.2 (`substitution_formula`)**.** Let $\mathcal{M}$ be an adequate model, w be a world, and $g, \tilde{g}$ be x-alternative w-assignments such that $\tilde{g}(x) = g(t)$. Then for every formula φ with t free for x:

$$\mathcal{M}, w \Vdash^{\tilde{g}} \varphi \iff \mathcal{M}, w \Vdash^{g} \varphi[x \leftarrow t].$$

Finally, the soundness of the constant elimination rule (Rule 3.1.(x), `ConstE`) is the trickiest, and we postpone its discussion to Section 5.2.

5.1. Finite Sets

We made a decision to only work with finite sets. This allowed us to make use of the nice Finite Maps library for choice types of MathComp [27] instead of having to prove many basic facts from scratch. However, Lemma 5.1 (above) made us momentarily reconsider this decision.

This lemma feels intuitive and in fact its proof was omitted in [1] and [5]. However, it is not as straightforward as it looks. A simple induction is underpowered to solve it; one must do induction building with the assumption that g and h are $(\mathrm{Vars} \setminus \mathrm{fv}(\varphi))$-alternative instead. Since $\mathrm{Vars} \setminus \mathrm{fv}(\varphi)$ is not a finite set, it can't be represented by the machinery of the Finite Maps library. In order to get around this, we defined the notion of Γ-equivalent assignments.

Definition 5.3 (`Xeq`)**.** Two w-assignments g and h are said to be Γ-equivalent if they agree on every variable in Γ.

Clearly g and h are $(\mathrm{Vars} \setminus \mathrm{fv}(\varphi))$-alternative if and only if they are $\mathrm{fv}(\varphi)$-equivalent. With this formulation we can prove Lemma 5.4 by induction first and obtain Lemma 5.1 as an easy corollary.

Lemma 5.4 (`sat_Xeqfv`)**.** Let $\mathcal{M}$ be an adequate model, w be a world, φ be a formula, and g, h be $\mathrm{fv}(\varphi)$-equivalent w-assignments. Then:

$$\mathcal{M}, w \Vdash^{g} \varphi \iff \mathcal{M}, w \Vdash^{h} \varphi.$$

5.2. Soundness of the Constant Elimination Rule

Recall the constant elimination rule (Rule 3.1.(x), `ConstE`):

$$\text{if } \varphi[x{\leftarrow}c] \rightsquigarrow \psi[x{\leftarrow}c], \text{ then } \varphi \rightsquigarrow \psi$$
$$(c \text{ not in } \varphi \text{ nor } \psi)$$

The argument for its soundness goes as follows. Suppose that $\varphi[x{\leftarrow}c] \rightsquigarrow \psi[x{\leftarrow}c]$ is sound and that $\mathcal{M}, w \Vdash^g \varphi$ for some adequate model $\mathcal{M}$, world w, and w-assignment g. We wish to show that $\mathcal{M}, w \Vdash^g \psi$. We build a new model $\mathcal{M}[w, c{\leftarrow}g(x)]$ that is identical to $\mathcal{M}$ except it interprets c as $g(x)$ in w, in hopes that $\mathcal{M}[w, c{\leftarrow}g(x)]$ satisfies $\chi[x{\leftarrow}c]$ if and only if $\mathcal{M}$ satisfies χ, for any formula χ where c does not appear. We can then deduce that $\mathcal{M}[w, c{\leftarrow}g(x)], w \Vdash \varphi[x{\leftarrow}c]$ from our assumption that $\mathcal{M}, w \Vdash^g \varphi$, and, since $\varphi[x{\leftarrow}c] \rightsquigarrow \psi[x{\leftarrow}c]$ is sound, this means that $\mathcal{M}[w, c{\leftarrow}g(x)], w \Vdash^g \psi[x{\leftarrow}c]$, and consequently that $\mathcal{M}, w \Vdash^g \psi$.

The above proof sketch should be intuitive enough, but it omits a crucial point: the naive definition of $\mathcal{M}[w, c{\leftarrow}g(x)]$ is not concordant, because the interpretation of a constant is being changed at w without being changed anywhere else. It is fine to propagate the change to the successors of w through the compatibility functions, and this would restore the concordance if w were the root of the model. However, when w is not the root, there is no clear solution other than dropping every other world from the model, which is what is done in [1]. It works well because the satisfaction of a formula at w depends only on the model restricted to w and its successors.

We originally tried to implement this proof directly: restrict $\mathcal{M}$ to w and its successors and then replace the interpretation of c with $g(x)$ at w and with $\eta_{w,u}(g(x))$ at all the successors u of w. In this proof, the models are adequate every step of the way. However, implementing this strategy proved rather difficult. A model restricted to w and its successors is naturally defined as a regular model together with a non-informative statement to the effect that every world is either w or its successor. Then the next step would be to define a way to change the interpretation of a constant at the root and propagate it to all its successors. However, trying to do this on top of restricted models proved hard, in part because there is no built-in concept of root. Adequate models do not need to be rooted and we didn't want to include this restriction.

Instead, we ended up changing the proof to postpone including non-informative elements as much as possible. The key insight is that only the final model needs to be adequate, and so we can change the constant interpretation first and only then restrict the worlds to obtain concordance. Here is also where the decision to have compatibility functions for every pair of worlds shines, as we'll soon see. We define the constant interpretation for the new model as follows.

Definition 5.5 (`replace_I`)**.** Let $\mathcal{M}$ be a model, w be a world, c be a constant, and d be an element of the domain of w. If I is the constant interpretation of $\mathcal{M}$, we define a new interpretation $I[w, c{\leftarrow}d]$ as follows. For a given world u, $c^{I[w,c\leftarrow d]_u} := \eta_{w,u}(d)$. $I[w, c{\leftarrow}d]$ behaves like I for every other constant.

Note that the above definition is well-typed even if $\mathcal{M}$ is not an adequate model, and it won't lead to an adequate model unless w happens to be the root of $\mathcal{M}$. Note also that, if $\mathcal{M}$ is adequate, then $c^{I[w,c\leftarrow d]_w} = \eta_{w,w}(d) = d$, because $\eta_{w,w}$ is the identity in adequate models.

Finally, observe that if $\eta_{w,u}$ only existed when wRu, we could not have defined $I[w, c{\leftarrow}d]$ like this, for there would be no way to obtain an element of the domain of u in the cases where u was not a successor of w. Recall that we're going to drop these worlds later anyway, so it doesn't matter which domain element this is; only that we have one in hand. Even though this could have been implemented in other ways (for example, by designating a default element for each domain), this particular solution is elegant in its simplicity and symmetry, as there is no need to have a case distinction on wRu.

The first approximation to $\mathcal{M}[w, c{\leftarrow}g(x)]$ is then a copy of $\mathcal{M}$ with the constant interpretation replaced by $I[w, c{\leftarrow}g(x)]$. We can already prove the desired property about this model (`sat_replace`), namely that $\mathcal{M}[w, c{\leftarrow}g(x)]$ satisfies $\chi[x{\leftarrow}c]$ at w if and only if $\mathcal{M}$ satisfies χ at w, for any formula χ where c does not appear. It now remains to further modify $\mathcal{M}[w, c{\leftarrow}g(x)]$ so that it is adequate, by dropping all spurious worlds, obtaining $\mathcal{M}[w, c{\leftarrow}g(x)]|_w$. The final model, called `restrict_replace`, is finally adequate and allows us to prove the desired result, Lemma 5.6, which has the soundness of the constant elimination rule as a corollary.

Lemma 5.6 (`sat_restrict_replace`)**.** Given a constant c, a formula φ where c does not appear, an adequate model $\mathcal{M}$, a world w, and a w-assignment g, we have:

$$\mathcal{M}, w \Vdash^g \varphi \iff \mathcal{M}[w, c{\leftarrow}g(x)]|_w, w \Vdash^g \varphi[x{\leftarrow}c].$$

6. Conclusions and Future Work

In this work we presented a Coq mechanization of the QRC_1 logic, its Kripke semantics, and a formalized proof of its soundness theorem. We discussed the difficulties in translating these objects and results to Coq as well as our proposed solutions. The formalization process suggested a slightly different definition of Kripke model that is nevertheless equivalent to the previous one under common assumptions. This new definition allowed for a simpler soundness proof.

The clear next step is to formalize Theorem 4.5, the completeness of QRC_1, possibly making use of Autosubst [34] to ease complications with binders. With both the axiom system and the completeness proof, it should be possible to generate a mechanized and formalized decision procedure for QRC_1 via Post's Theorem, which has already been formalized itself [38]. It would also be interesting to see if some of the techniques described in the recent formalization of a decision procedure for GL in HOL Light [15] are applicable, since GL is a closely related logic.

Other modal results on QRC_1 could be formalized too, such as the fact that it is the strictly positive fragment of the quantified modal logics between $\mathsf{QK4}$ and QGL [5]. Finally, the arithmetical results could be an interesting subject, although these are less elementary and would need to be part of a larger project including practical definitions of arithmetical theories such as Peano Arithmetic and its fragments. The Undecidability Library [38] might be a good basis for such a project.

Acknowledgments

The author wishes to thank the Coq community for its ready support during the formalization process, Joost J. Joosten for proposing this article could be written, Mireia González Bedmar for

her comments on an early draft, and the anonymous reviewers for their helpful suggestions.

References

[1] A. de Almeida Borges, J. J. Joosten, Quantified Reflection Calculus with one modality, in: N. Olivetti, R. Verbrugge, S. Negri, G. Sandu (Eds.), Advances in Modal Logic 13, College Publications, 2020, pp. 13–32.

[2] E. V. Dashkov, On the positive fragment of the polymodal provability logic GLP, Mathematical Notes 91 (2012) 318–333.

[3] L. D. Beklemishev, Calibrating provability logic: From modal logic to Reflection Calculus, in: T. Bolander, T. Braüner, T. S. Ghilardi, L. Moss (Eds.), Advances in Modal Logic 9, College Publications, London, 2012, pp. 89–94.

[4] V. A. Vardanyan, Arithmetic complexity of predicate logics of provability and their fragments, Doklady Akad. Nauk SSSR 288 (1986) 11–14. In Russian. English translation in Soviet Mathematics Doklady 33, 569–572 (1986).

[5] A. de Almeida Borges, J. J. Joosten, An escape from Vardanyan's Theorem, The Journal of Symbolic Logic (2022). URL: https://www.cambridge.org/core/journals/journal-of-symbolic-logic/article/abs/an-escape-from-vardanyans-theorem/03E13D4C282563F918AF77CF6E2830DA. doi:10.1017/jsl.2022.38.

[6] A. de Almeida Borges, Coq formalization of QRC_1, 2022. URL: https://doi.org/10.5281/zenodo.6615336. doi:10.5281/zenodo.6615336.

[7] R. Goldblatt, Quantifiers, propositions and identity, admissible semantics for quantified modal and substructural logics, Cambridge University Press, 2011.

[8] D. Basin, S. Matthews, L. Viganò, Modal logics: Quantifiers, Journal of Logic, Language and Information 7 (1998) 237–263.

[9] C. Benzmüller, B. Woltzenlogel Paleo, Interacting with modal logics in the Coq Proof Assistant, in: L. D. Beklemishev, D. V. Musatov (Eds.), Computer Science – Theory and Applications, Springer International Publishing, Cham, 2015, pp. 398–411.

[10] T. Libal, A simple semi-automated proof assistant for first-order modal logics, in: C. Benzmüller, J. Otten (Eds.), Proceedings of the 3rd International Workshop on Automated Reasoning in Quantified Non-Classical Logics (ARQNL 2018), 2018, pp. 34–48.

[11] T. Gleißner, A. Steen, C. Benzmüller, Theorem provers for every normal modal logic, in: T. Eiter, D. Sands (Eds.), LPAR-21: 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning, volume 46 of *EPiC Series in Computing*, 2017, pp. 14–30.

[12] C. Doczkal, G. Smolka, Constructive formalization of hybrid logic with eventualities, in: Z. S. Jean-Pierre Jouannaud (Ed.), Certified Programs and Proofs, (CPP 2011), volume 7086 of *LNCS*, Springer, 2011, pp. 5–20.

[13] C. Doczkal, J. Bard, Completeness and Decidability of Converse PDL in the Constructive Type Theory of Coq, in: Certified Programs and Proofs, (CPP 2018), Los Angeles, United States, 2018, pp. 42–52.

[14] A. de Almeida Borges, Worms in Coq, 2018. URL: https://gitlab.com/ana-borges/WormsCoq.

[15] M. Maggesi, C. P. Brogi, A theorem prover and countermodel constructor for provability logic in HOL Light, 2022. arXiv:2205.03659 [cs.LO].
[16] R. O'Connor, Essential incompleteness of arithmetic verified by Coq, in: J. Hurd, T. Melham (Eds.), Proceedings of the 18th international conference on Theorem Proving in Higher Order Logics, volume 3603 of *Theoretical Computer Science and General Issues*, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 245–260.
[17] Y. Forster, D. Kirst, D. Wehr, Completeness theorems for first-order logic analysed in constructive type theory: extended version, Journal of Logic and Computation 31 (2021) 112–151.
[18] The Coq Development Team, The Coq Proof Assistant, 1989. URL: http://coq.inria.fr.
[19] M. Sozeau, S. Boulier, Y. Forster, N. Tabareau, T. Winterhalter, Coq Coq Correct! Verification of type checking and erasure for Coq, in Coq, Proceedings of the ACM on Programming Languages 4 (2020).
[20] G. Gonthier, Formal proof – the four-color theorem, Notices of the American Mathematical Society 55 (2008) 1382–1393.
[21] G. Gonthier, A. Asperti, J. Avigad, Y. Bertot, C. Cohen, F. Garillot, S. Le Roux, A. Mahboubi, R. O'Connor, S. Ould Biha, I. Pasca, L. Rideau, A. Solovyev, E. Tassi, L. Théry, A machine-checked proof of the Odd Order Theorem, in: S. Blazy, C. Paulin-Mohring, D. Pichardie (Eds.), Interactive Theorem Proving, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 163–179.
[22] T. Hales, M. Adams, G. Bauer, T. D. Dang, J. Harrison, L. T. Hoang, C. Kaliszyk, V. Magron, S. McLaughlin, T. T. Nguyen, et al., A formal proof of the Kepler Conjecture, Forum of Mathematics, Pi 5 (2017) 1–29.
[23] C. Doczkal, G. Combette, D. Pous, A formal proof of the minor-exclusion property for treewidth-two graphs, in: J. Avigad, A. Mahboubi (Eds.), ITP 2018: Interactive Theorem Proving, volume 10895 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2018, pp. 178–195. Coq code: http://perso.ens-lyon.fr/christian.doczkal/itp18/index.html.
[24] X. Leroy, Formal verification of a realistic compiler, Commun. ACM 52 (2009) 107–115.
[25] A. de Almeida Borges, M. González Bedmar, J. Conejero Rodríguez, E. Hermo Reyes, J. Casals Buñuel, J. J. Joosten, FV Time: a formally verified Coq library, 2022. URL: https://arxiv.org/abs/2209.14227, arXiv:2209.14227 [cs.SE].
[26] The Mathematical Components Team, The Mathematical Components library, 2007. URL: https://math-comp.github.io/.
[27] C. Cohen, K. Sakaguchi, Finite maps, 2015. URL: https://github.com/math-comp/finmap.
[28] G. Gonthier, A. Mahboubi, E. Tassi, A Small Scale Reflection Extension for the Coq system, Research Report RR-6455, Inria Saclay Ile de France, 2016. URL: https://hal.inria.fr/inria-00258384.
[29] U. Kohlenbach, Applied Proof Theory: Proof Interpretations and their Use in Mathematics, Springer, Berlin, 2008.
[30] A. Chlipala, Certified programming with dependent types: a pragmatic introduction to the Coq Proof Assistant, MIT Press, 2013.
[31] A. Mahboubi, E. Tassi, Mathematical Components, Zenodo, 2021. doi:10.5281/zenodo.4457887.
[32] F. Wiedijk, The de Bruijn Factor, 2000. URL: https://www.cs.ru.nl/~freek/factor/factor.pdf,

accessed December 2022.

[33] N. G. de Bruijn, Lambda Calculus notation with nameless dummies: A tool for automatic formula manipulation, with application to the Church-Rosser Theorem, Indagationes Mathematicae 34 (1972) 381–392.

[34] K. Stark, S. Schäfer, J. Kaiser, Autosubst 2: Reasoning with multi-sorted de Bruijn terms and vector substitutions, 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14-15, 2019 (2019).

[35] A. Sernadas, C. Sernadas, Foundations of Logic and Theory of Computation, volume 10 of *Texts in Computing*, second ed., College Publications, London, 2012.

[36] K. Sakaguchi, Validating mathematical structures, in: N. Peltier, V. Sofronie-Stokkermans (Eds.), Automated Reasoning, Springer International Publishing, Cham, 2020, pp. 138–157.

[37] C. Cohen, K. Sakaguchi, E. Tassi, Hierarchy Builder: algebraic hierarchies made easy in Coq with Elpi, in: FSCD 2020 - 5th International Conference on Formal Structures for Computation and Deduction, number 167 in 5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020), Paris, France, 2020, pp. 34:1–34:21.

[38] Y. Forster, D. Larchey-Wendling, A. Dudenhefner, E. Heiter, M. Hermes, D. Kirst, M. Koch, F. Kunze, G. Smolka, S. Spies, D. Wehr, M. Wuttke, Coq library of undecidability proofs, 2018. URL: https://github.com/uds-psl/coq-library-undecidability.