Abstract, Compositional Consistency: Isabelle/HOL Locales for Completeness à la Fitting

Asta Halkjær From
Department of Computer Science, University of Copenhagen, Copenhagen, Denmark

Anders Schlichtkrull
Department of Computer Science, Aalborg University Copenhagen, Copenhagen, Denmark

Abstract

Smullyan and Fitting have used abstract consistency properties to great effect in unifying meta-theoretical results in logic. In this paper, we generalize these developments with the help of Isabelle/HOL. We use locales to decompose abstract consistency into general parts, and provide the textbook variants as special cases. Users can assemble their own consistency property for a given logic. The compositionality alleviates the absence of dependent types in Isabelle/HOL. We use our development to mechanize completeness of calculi for three logics: (1) first-order logic where we only instantiate universal quantifiers with already occurring terms, (2) second-order logic over general models, and (3) a recently developed strong hybrid logic with propositional quantification.

2012 ACM Subject Classification Theory of computation → Proof theory; Theory of computation → Higher order logic

Keywords and phrases Logic, completeness, abstract consistency property, Isabelle/HOL, locales

Supplementary Material *Mechanization*: `https://github.com/astahfrom/Analytic_Completeness` archived at `swh:1:dir:1bea24a089edbdbc233081a23b0dc45963bf6c8f`

Funding *Asta Halkjær From*: This work is supported by a Novo Nordisk Fonden Start Package grant (NNF20OC0063462).

Acknowledgements We thank the anonymous reviewers and Jørgen Villadsen for their comments.

1 Introduction

We often raise the question of completeness: do we have the necessary syntactic machinery to prove all semantic validities? One key technique for such proofs is the construction of a *canonical model* that falsifies underivable formulas, contradicting their validity. Such models can be built, in the Lindenbaum-Henkin tradition [25], by extending *consistent* sets into *maximal consistent sets* which include every formula consistent with them. We can define consistency with respect to a specific calculus, e.g., by saying that no contradiction can be derived via the rules of the proof system. However, Smullyan [38] gave what he called a "unifying principle in quantification theory" with his *abstract consistency properties* that open the door to more results than only completeness.

1.1 Abstract Consistency

The concept relies on Smullyan's unifying notation [38], introduced in the same paper, where formulas and their negations are classified as either α-, β-, γ- or δ-forms. The different forms relate to their respective sub-forms in different ways. The α-forms act conjunctively: $\phi \wedge \psi$ has α-form with sub-forms $\alpha_1 = \phi$ and $\alpha_2 = \psi$, but $\neg(\phi \vee \psi)$ also has α-form, with $\alpha_1 = \neg\phi$ and $\alpha_2 = \neg\psi$. The β-forms act disjunctively: $\phi \vee \psi$, $\phi \to \psi$, and $\neg(\phi \wedge \psi)$ all have β-form. The γ-forms act universally, as exemplified by the universal quantifier $\forall x.\, \phi(x)$ or the negated existential $\neg(\exists x.\, \phi(x))$, where the sub-forms $\gamma(t)$ are (possibly negated) instances of the

quantified formulas. Finally, the δ-forms act existentially: $\exists x.\, \phi(x)$ and $\neg(\forall x.\, \phi(x))$, again with instances, $\delta(t)$, as sub-forms.

The unified notation allows Smullyan to state consistency abstractly and succinctly by introducing consistency properties. Intuitively, a consistency property C is a family of sets of formulas $S \in C$ closed under conditions that ensure that each member set S is non-contradictory when understood as a (possibly infinite) conjunction. For instance, at most one of a propositional letter, A, and its negation, $\neg A$, can appear in a member set $S \in C$. Likewise, the consistency of a conjunction in a member set, $\phi \wedge \psi \in S \in C$, must be evidenced by the consistency of the conjuncts together with the member set, $\{\phi, \psi\} \cup S \in C$. The idea is that the member set $S \in C$ is justified by another member set, namely $\{\phi, \psi\} \cup S \in C$. In this way, consistency is eventually justified at the level of atomic formulas. The following formulation is adapted from Fitting's textbook [15] on first-order logic.

▶ **Definition 1** (Abstract Consistency). *A family of formula sets C is a first-order* consistency property *given that all sets $S \in C$ satisfy the following* kinds *of conditions.*[1]
conflict *for any propositional letter A, not both $A \in S$ and $\neg A \in S$.*
banned $\bot \notin S$; $\neg\top \notin S$.
double negation *if $\neg\neg Z \in S$ then $\{Z\} \cup S \in C$*
alpha *if $\alpha \in S$ then $\{\alpha_1, \alpha_2\} \cup S \in C$*
beta *if $\beta \in S$ then $\{\beta_1\} \cup S \in C$ or $\{\beta_2\} \cup S \in C$*
gamma *if $\gamma \in S$ then $\{\gamma(t)\} \cup S \in C$ for every closed term t of L^{par}*
delta$_E$ *if $\delta \in S$ then $\{\delta(p)\} \cup S \in C$ for some parameter p of L^{par}*

Given the above, Smullyan [38, 39] and Fitting [15] each prove variations of the key *model existence theorem*: Given a first-order consistency property C (over L) any set $S \in C$ of *sentences* is satisfiable (in a Herbrand model over L^{par}). The proof relies crucially on imposing *finite character* on the family C, i.e., $\forall S.\ S \in C \iff (\forall S' \subseteq S.\ \textit{finite}\ S' \longrightarrow S' \in C)$.

Given the axiom of choice, which we assume when working in Isabelle/HOL, finite character guarantees the existence of a maximal element with respect to set inclusion. We want to build our canonical model from exactly this maximal consistent set, by showing that it has various *Hintikka properties*. A Hintikka set H justifies its own consistency. For example, the Hintikka property corresponding to **alpha** says that if $\alpha \in H$, then $\alpha_1 \in H$ and $\alpha_2 \in H$ as well. For **beta**, we have that if $\beta \in H$, then $\beta_1 \in H$ or $\beta_2 \in H$.

Unfortunately, imposing finite character does not necessarily preserve the **delta$_E$** kind as given above. Instead, we must use the following *alternate* kind that quantifies universally over all new parameters:
delta$_A$ if $\delta \in S$ then $\{\delta(p)\} \cup S \in C$ for every parameter p new to S

The idea is then to recover satisfaction of the original **delta$_E$** kind by manually inserting witnesses during the construction of the maximal consistent set with Lindenbaum's lemma.

1.2 Applications

The applications of the model existence theorem are many, which is why Smullyan called it a unifying principle. Gödel's completeness theorem for a given (first-order) proof system follows immediately by proving that the concrete consistency of the proof system constitutes an abstract consistency property. The same model existence result can be reused for different

[1] The language L^{par} extends the original language L with infinitely many *parameters* (constant symbols).

proof systems. Notably, the conditions in Definition 1 all obey the sub-formula property, so we do not need to include cut rules in the proof systems. This is why Smullyan also called these *analytic* consistency properties [39]. Gentzen's Hauptsatz, the cut-elimination theorem, is an immediate consequence. The compactness theorem follows by constructing a consistency property from all sets of sentences where every finite subset is satisfiable [15]. A weak version of the downward Löwenheim-Skolem theorem follows by restricting ourselves to a countable language, building a consistency property from all satisfiable sets of sentences, and noticing that the domain of the Herbrand model is also countable [15]. Craig's interpolation theorem follows by proving that the sets with an interpolant-free partition constitute a consistency property [15]. In all cases, we need to restrict ourselves to sets that leave enough parameters new, so that we can always witness δ-forms.

1.3 Isabelle/HOL

Approximately 5000 nonblank lines of formal text in Isabelle/HOL 2025 supplement this paper. We include parts of it here via Isabelle/HOL's LaTeX export (which renders underscores as hyphens). We use the following symbols to denote different kinds of snippets:

△ *Triangle denotes a term by its assumed type or given definition.*

○ *Circle denotes the assumption of a lemma or theorem.*

∴ *Three dots denote the corresponding conclusion.*

∗ *The asterisk denotes a proof obligation in a given context, typically a locale.*

We write type variables as $'a$ and function types with $\Rightarrow$. We annotate types with ::. We often prefer Isabelle's meta-logical universal quantifier, $\bigwedge$, and implication, $\Longrightarrow$, to their object-logic counterparts $\forall$ and $\longrightarrow$. We apply functions and predicates without parentheses, e.g., $f\ x\ y$, and make ample use of anonymous functions $\lambda x.\ e$. The *range* of a function is its image. The image of a function f on a set S is denoted $f\ `\ S$. We write lists enclosed in square brackets, [...], and use *map* to apply a function to every element, and *set* to get the set of elements. Sets are potentially infinite, and we write union ($\cup$), membership ($\in$), etc., as usual. The polymorphic term *UNIV* denotes the set of all elements of a type. We refer to Nipkow et al.'s book [33] for a thorough introduction.

We make heavy use of Isabelle/HOL's locales [2]. Locales allow us to abstract over definitions subject to various properties and to build a hierarchy of these. They are key to generalizing consistency without losing user friendliness. For all their benefits, locales are still subject to the limitations of HOL and do not give us quantification over type variables. But, as we show, we can instead compose locales instantiated at the types we care about.

1.4 Contributions

Fitting [16] extended the notion of abstract consistency property to cover term-modal logics with the following kind of condition:

modal if $\Diamond_t \phi \in S$ then $\{\phi\} \cup \{\psi \mid \Box_t \psi \in S\} \in C$

The details here are not important. However, the condition has a different shape to Smullyan's original ones: Fitting does not just extend the set S with a sub-formula but transforms S entirely with a set comprehension. Fitting [14] uses similar conditions to handle intuitionistic logics. This raises a central question of this paper:

How abstract can we make abstract consistency properties?

The kinds of conditions we have seen so far have several flavors: **conflict** outlaws certain formulas conditionally, while **banned** forbids them outright; **double negation** and **alpha** postulate that certain larger sets must also be consistent; **beta** postulates that *at least one* of two larger sets are consistent; **gamma** has an infinitary character; **delta$_\mathsf{E}$** and **delta$_\mathsf{A}$** draw on a set of parameters and are intimately tied to the construction of the maximal set; finally **modal** transforms the given set entirely. In the end, what matters is that these properties are preserved when we give the family of sets finite character.

In this paper, we mechanize an abstract consistency *kind* that generalizes all of the above, and build consistency properties over any finite set of such kinds. We implement the textbook definitions as special cases with the following generalizations:

- Some proof systems use gadgets such as labels. To accommodate this, we let the user choose what their "formulas" look like.
- We do not restrict ourselves to sentences.
- We do not restrict ourselves to conditioning on single formulas, but can treat finite sets of formulas as α-, β-, and so on, -forms. This makes it easier to work with proof systems with rules contingent on multiple formulas.
- We generalize **gamma** to any notion of quantifier and any notion of term, including formulas themselves. Our second-order and hybrid logic examples rely crucially on this.
- We let the universe of terms that **gamma** quantifies over depend on the set of formulas that contains the γ-form. Our first-order example demonstrates that we only need to instantiate quantifiers with terms that already occur in the derivation.
- We support uncountable languages.

All of this is enabled by the *locale* feature [2] of Isabelle/HOL, which allows us to work very abstractly. In particular, we support as varied clauses as the following. For first-order logic (FOL), we evidence the universal quantifier by its instances over the *terms* at hand:

△ $[\,\forall\, p\,] \rightsquigarrow_\gamma (\mathit{terms}, \lambda t.\ [\,\langle t\rangle p\,])$

Here, $\forall\, p$ binds de Bruijn variable 0 and $\langle t\rangle p$ instantiates it with the term t.

For second-order logic (SOL), we can quantify over first-order terms, function symbols, and predicate symbols, and have three clauses to match, each with its own instantiation:

△ $[\,\forall\, p\,] \rightsquigarrow_\gamma (\lambda t.\ [\,\langle t/0\rangle p\,])$

△ $[\,\forall_F\ p\,] \rightsquigarrow_{\gamma F} (\lambda s.\ [\,\langle s/0\rangle_F\ p\,])$

△ $[\,\forall_P\ p\,] \rightsquigarrow_{\gamma P} (\lambda s.\ [\,\langle s/0\rangle_P\ p\,])$

Again, $\forall\, p$ binds de Bruijn variable 0 and $\langle t/0\rangle p$ instantiates it with the term t.

For Prior's Ideal Language (PIL) [4], a strong hybrid logic with propositional quantification, we use labels i, k to name the point of evaluation, and these participate freely in our kinds:

△ $[\,(i,\ \Box p),\ (i,\ \Diamond(\bullet k))\,] \rightsquigarrow_\alpha [(k,\ p)]$

△ $[\,(i,\ \mathbf{A}\ p)\,] \rightsquigarrow_{\gamma i} (\lambda k.\ [\,(k,\ p)\,])$

The first clause states that if p holds everywhere we look, and we can see world k, then p holds at k. The second clause states that if p holds globally, then p holds at all labels k.

The ability to include more than one **gamma** kind at a time alleviates the absence of dependent types: each **gamma** can only quantify over one type of terms, but we can simply build our consistency property from multiple **gamma**s. This once again demonstrates that *simple type theory is not too simple* [9].

In total, our contributions are:

- Abstract machinery for building maximal consistent sets with a wide range of applications.

- Generalized versions of the textbook consistency kinds, pre-defined as special cases using Isabelle/HOL's locales.
- Completeness proofs for various proof systems for three different logics: first-order logic, second-order logic, and hybrid logic with propositional quantification.
- Compactness for first-order logic over a language of any cardinality.

However, we cannot use different type variables to represent, say, first-order constant symbols and second-order function symbols if we want **delta$_\mathsf{E}$** kinds for both.

2 Abstract Consistency

We outline the overall strategy before diving into its mechanization. We seek to characterize the consistency of families of sets, build a maximal element, and guarantee properties useful for constructing a model. Smullyan [38, 39] and Fitting [15, 16] have already developed the theory of abstract consistency properties (cf. Definition 1), and we can build on their work. We follow Fitting's [15] strategy for building the maximal element. Notably, we have not fixed a semantics, so we cannot actually construct a model (we leave this to our examples), but we can provide all the structure needed for doing so.

Lindenbaum's lemma will do most of the work, but we first need to give the family of sets C finite character to guarantee that the maximal element exists. That is, a set that satisfies:

△ $\mathit{maximal}\ C\ S \longleftrightarrow (\forall S' \in C.\ S \subseteq S' \longrightarrow S = S')$

We cannot impose full finite character on a consistency property without disrupting the **delta$_\mathsf{E}$** kind, but we can actually close it under subsets without issues. This motivates Fitting's three-step process for imposing finite character on a family of sets C [15]:

1. Subset close the family (giving it downward finite character).
2. Satisfy **delta$_\mathsf{A}$** by adding all sets that have a consistent parameter substitution.
3. Give the family (upward) finite character.

Step 3 destroys satisfaction of the existential **delta$_\mathsf{E}$** kind, but preserves it for the **delta$_\mathsf{A}$** kind satisfied by step 2. Fitting [15, 16] thus proves, for his notions of consistency, that the above process results in an *alternate consistency property* of finite character.

To construct his model, Fitting notes that the maximal consistent set H has a number of *Hintikka* properties corresponding to the consistency conditions in Definition 1. The **conflict** and **banned** conditions simply apply to H, but, say, **alpha** now simplifies to the property that if $\alpha \in H$ then $\{\alpha_1, \alpha_2\} \subseteq H$, since H is a maximal element. Similarly for the rest of the conditions (for **modal** [16] it depends on the logic in question). These Hintikka properties of the maximal element of H are exactly what make it so suitable for constructing a model.

2.1 Parameter Substitutions

The outline above tells us that parameter substitutions are essential to our development.

▶ **Definition 2** (Parameter Substitutions). *We work with any type 'x of parameters and 'fm of formulas given the following operations. Formulas must have finite sets of parameters, the identity substitution must be the identity function, and substitutions must act equally on a formula when they agree on the parameters of that formula:*

△ $\mathit{map\text{-}fm} :: ('x \Rightarrow 'x) \Rightarrow 'fm \Rightarrow 'fm$

△ $\mathit{params\text{-}fm} :: 'fm \Rightarrow 'x\ \mathit{set}$

∗ $\bigwedge p.\ \mathit{finite}\ (\mathit{params\text{-}fm}\ p)$

∗ $\mathit{map\text{-}fm}\ \mathit{id} = \mathit{id}$

∗ $\bigwedge f\ g\ p.\ (\forall x \in \mathit{params\text{-}fm}\ p.\ f\ x = g\ x) \Longrightarrow \mathit{map\text{-}fm}\ f\ p = \mathit{map\text{-}fm}\ g\ p$

In many instances, Isabelle/HOL's datatype package [6] can automatically generate such operations. By building all our locales on top of this locale, we use the same parameter substitution throughout the development, and users need only prove these properties once.

2.2 Abstract Kinds

We want to generalize properties as varied as those we saw in Definition 1: **conflict**, **alpha**, **beta**, etc., and **modal** too. At the same time, we need to be able to impose finite character on any set subject to these different kinds of conditions.

We note that each kind of condition in a consistency property only applies to a set $S \in C$ when S contains a specific type of formula, and that when it does, this imposes a restriction on S and C specific to that formula. For example, the **alpha** kind associates α-forms with conditions $\{\alpha_1, \alpha_2\} \cup S \in C$, and says nothing about sets without α-forms. However, it does specialize to a Hintikka property on the maximal set. We also need the ability to interpret the (user's choice of) δ-forms in two different ways: existentially as in $\mathbf{delta_E}$, and universally as in $\mathbf{delta_A}$. This motivates our datatype of kinds.

▶ **Definition 3** (Abstract Kinds). *A kind K is a datatype (of type ('x, 'fm) kind) with two variants. Variant **Cond** consists of two predicates of types 'fm list ⇒ ('fm set set ⇒ 'fm set ⇒ bool) ⇒ bool and 'fm set ⇒ bool, respectively. Variant **Wits** consists of a function of type 'fm ⇒ 'x ⇒ 'fm list.*

Conditions have two components: (1) a relation between lists of (contained) formulas and conditions on C and $S \in C$, and (2) a Hintikka property. Witnessings take a formula and a parameter and produce a (possibly empty) list of witnesses. With this definition, we see that a family of sets C can satisfy a kind in three different ways.

▶ **Definition 4** (Kind Satisfaction). *Take a family of sets C and a kind K. We define sat_E of type ('x, 'fm) kind ⇒ 'fm set set ⇒ bool, sat_A of type ('x, 'fm) kind ⇒ 'fm set set ⇒ bool and sat_H of type ('x, 'fm) kind ⇒ 'fm set ⇒ bool as follows.*

The family C satisfies K existentially *when (Cond) any condition Q imposed by a list of formulas ps from $S \in C$ holds for C and S, and (Wits) some parameter is used as witness:*

△ $(\bigwedge S\ ps\ Q.\ S \in C \Longrightarrow set\ ps \subseteq S \Longrightarrow P\ ps\ Q \Longrightarrow Q\ C\ S) \Longrightarrow sat_E\ (Cond\ P\ H)\ C$

△ $(\bigwedge S\ p.\ S \in C \Longrightarrow p \in S \Longrightarrow (\exists x.\ set\ (W\ p\ x) \cup S \in C)) \Longrightarrow sat_E\ (Wits\ W)\ C$

The family C satisfies K universally *(or alternately) when (Cond) as above, and (Wits) all new parameters are used as witnesses:*

△ $(\bigwedge S\ ps\ Q.\ S \in C \Longrightarrow set\ ps \subseteq S \Longrightarrow P\ ps\ Q \Longrightarrow Q\ C\ S) \Longrightarrow sat_A\ (Cond\ P\ H)\ C$

△ $(\bigwedge S\ p\ x.\ S \in C \Longrightarrow p \in S \Longrightarrow x \notin params\ S \Longrightarrow set\ (W\ p\ x) \cup S \in C) \Longrightarrow$
$sat_A\ (Wits\ W)\ C$

The set of formulas S satisfies K as a Hintikka *kind when (Cond) the set fulfills the Hintikka property given by the user, and (Wits) the witnesses for some parameter are included:*

△ $H\ S \Longrightarrow sat_H\ (Cond\ P\ H)\ S$

△ $(\bigwedge p.\ p \in S \Longrightarrow (\exists x.\ set\ (W\ p\ x) \subseteq S)) \Longrightarrow sat_H\ (Wits\ W)\ S$

▶ Remark 5. We use Isabelle/HOL's *inductive* command to specify the various predicates above. We could also have used the *primrec* command. However, we found that it takes very little work with *inductive*, *inductive-cases* and judicious use of *intro* and *elim* attributes, to make our abstractions "melt away" for concrete kinds, making our generality very cheap.

We lift the above interpretations to lists of kinds in the expected way.

▶ **Definition 6** (Properties). *The regular/existential consistency properties, alternate/universal consistency properties, and Hintikka properties characterized by a list of kinds are:*
△ $prop_E\ Ks\ C \equiv \forall K \in set\ Ks.\ sat_E\ K\ C$
△ $prop_A\ Ks\ C \equiv \forall K \in set\ Ks.\ sat_A\ K\ C$
△ $prop_H\ Ks\ S \equiv \forall K \in set\ Ks.\ sat_H\ K\ S$

As outlined earlier, not all kinds suit our needs: the following locale carves out the *consistency kinds* that behave well with respect to giving the family of sets finite character. To state it, we first need to mechanize the operations needed for the three-step strategy:
△ $close\ C \equiv \{S.\ (\exists S' \in C.\ S \subseteq S')\}$ *and* $subset\text{-}closed\ C \equiv \forall S' \in C.\ \forall S \subseteq S'.\ S \in C$
△ $mk\text{-}alt\text{-}consistency\ C \equiv \{S.\ (\exists f.\ map\text{-}fm\ f\ `\ S \in C)\}$
△ $mk\text{-}finite\text{-}char\ C \equiv \{S.\ (\forall S' \subseteq S.\ finite\ S' \longrightarrow S' \in C)\}$

▶ **Definition 7** (Consistency Kinds). *A kind K is a* consistency kind *when it supports the imposition of finite character, and when its Hintikka condition holds for the maximal set:*
∗ $\bigwedge C.\ sat_E\ K\ C \Longrightarrow sat_E\ K\ (close\ C)$
∗ $\bigwedge C.\ sat_E\ K\ C \Longrightarrow subset\text{-}closed\ C \Longrightarrow sat_A\ K\ (mk\text{-}alt\text{-}consistency\ C)$
∗ $\bigwedge C.\ subset\text{-}closed\ C \Longrightarrow sat_A\ K\ C \Longrightarrow sat_A\ K\ (mk\text{-}finite\text{-}char\ C)$
∗ $\bigwedge C\ S.\ sat_E\ K\ C \Longrightarrow S \in C \Longrightarrow maximal\ C\ S \Longrightarrow sat_H\ K\ S$

▶ **Lemma 8.** *All witnessing kinds trivially satisfy their Hintikka condition:*
∴ $sat_E\ (Wits\ W)\ C \Longrightarrow S \in C \Longrightarrow maximal\ C\ S \Longrightarrow sat_H\ (Wits\ W)\ S$

We now have all the structure we need for our first central theorem.

▶ **Theorem 9** (Consistency Kinds Respect Finite Character). *Given a list of* consistency kinds *Ks, we can transform any consistency property C into an alternate consistency property of finite character that includes all sets from C:*
△ $mk\text{-}alt\text{-}fin\ C \equiv mk\text{-}finite\text{-}char\ (mk\text{-}alt\text{-}consistency\ (close\ C))$
∴ $finite\text{-}char\ (mk\text{-}alt\text{-}fin\ C)$
∴ $S \in C \Longrightarrow S \in mk\text{-}alt\text{-}fin\ C$
∴ $prop_E\ Ks\ C \Longrightarrow prop_A\ Ks\ (mk\text{-}alt\text{-}fin\ C)$

2.3 Model Existence

From here, Lindenbaum's construction gives us a way to extend an element S of an alternate consistency property C of finite character into a maximal element. We need a well-ordering of formulas and the assumption that there are enough parameters new to S to ensure consistency of the witnessings. In this more abstract setting than Smullyan and Fitting's countable infinity, having enough new parameters means that the cardinality of new parameters is at least the cardinality of our universe of formulas.
△ $enough\text{-}new\ S \equiv |UNIV :: 'fm\ set| \leq_o |- params\ S|$

▶ Remark 10. For countable types, it suffices to have infinitely many new parameters available.
○ $\exists\ to\text{-}nat :: 'fm \Rightarrow nat.\ inj\ to\text{-}nat$
○ $infinite\ (- params\ S)$
∴ $enough\text{-}new\ S$

We want to be clear that the *enough-new* predicate gives a more fine-grained account than extending the language with additional constants, but it does not preclude doing so.

We enumerate formulas by assuming an infinite cardinal order on their universe. Given infinitely many formulas, we can always impose one using the axiom of choice. We use well order recursion [7] to enumerate the consistent sets regardless of the language's cardinality.

▶ **Definition 11** (Lindenbaum). *The union of an enumeration of consistent sets gives us the maximal extension of $S \in C$. The enumeration, extend C S a, has three cases depending on formula a's place in the well order. For zero elements, take S; for successor elements, extend S with a if this preserves consistency (and add witnesses for a); for limit elements, take the union of all sets at strictly smaller elements:*

△ $Extend\ C\ S \equiv \bigcup a.\ extend\ C\ S\ a$

△ $extend\ C\ S\ a \equiv worecZSL\ S\ (extendS\ C)\ (extendL\ C)\ a$

△ $extendS\ C\ a\ prev \equiv if\ (\{a\} \cup prev \in C)\ then\ (witness\ a\ prev \cup \{a\} \cup prev)\ else\ prev$

△ $extendL\ C\ rec\ a \equiv \bigcup b \in underS\ a.\ rec\ b$

We *witness* a formula p with respect to a list of kinds Ks by applying any witnessings in Ks to p and a new parameter chosen by Hilbert's choice operator.

▶ **Lemma 12.** *Witnessing recovers the existential treatment of δ-forms:*

∴ $Wits\ W \in set\ Ks \Longrightarrow \exists x.\ set\ (W\ p\ x) \subseteq witness\ p\ S$

To build the maximal element, we apply the Lindenbaum construction at the transformed consistency property:

△ $mk\text{-}mcs\ C\ S \equiv Extend\ (mk\text{-}alt\text{-}fin\ C)\ S$

The maximal element is consistent, maximal, and *witnessed* (alternatively δ-complete). Inclusion in the (transformed) consistency property follows from its finite character. We can prove maximality because we closed the consistency property under subsets. Finally, the witnessing follows from the construction. These proofs are all available in the supplementary material. The following theorem consolidates them.

▶ **Theorem 13** (The Maximal Consistent Set is Hintikka). *Assume that C is a consistency property, that $S \in C$, and that there are enough parameters new to S to consistently construct all witnesses. Then the maximal consistent set is Hintikka:*

∘ $prop_E\ Ks\ C$

∘ $S \in C$

∘ $enough\text{-}new\ S$

∴ $prop_H\ Ks\ (mk\text{-}mcs\ C\ S)$

3 Concrete Consistency

The previous section developed abstract consistency properties in the abstract. In this section, we demonstrate that our abstractions are sound by showing that we can recover Smullyan and Fitting's original definitions. In fact, our foundation enables us to generalize them, as described in the introduction, making them even more useful.

Again, we rely crucially on Isabelle/HOL's locale feature. This time, we use sub-locales to provide a more readily applicable interface to our development than our foundational locale. Smullyan and Fitting have already identified a range of useful kinds, and we define each of these as a specific instance of our *consistency kinds*. This gives users a choice: use the base locale for maximal freedom at the cost of having to prove everything manually, or use our sub-locales, with a more restricted interface, but where the proofs are already given.

3.1 Conflicts, Alpha, and Beta

These first three kinds are very similar. Our flexible setup allows us to express **conflict** and **banned** from Definition 1 in one notion. Our pre-defined kinds all build on a user-provided relation between a list of formulas and its "consequence", from which we derive the concrete kind. For *Confl*, *Alpha*, and *Beta*, we simply relate two lists of formulas, and use the infix notations: $\rightsquigarrow_{\boldsymbol{\times}}$, $\rightsquigarrow_\alpha$, and $\rightsquigarrow_\beta$, respectively. The user must prove, as the only thing, that parameter substitution preserves the relation. For instance for conflicts:

$*$ $\bigwedge ps\ qs\ f.\ ps \rightsquigarrow_{\boldsymbol{\times}} qs \Longrightarrow map\ (map\text{-}fm\ f)\ ps \rightsquigarrow_{\boldsymbol{\times}} map\ (map\text{-}fm\ f)\ qs$

We can then define the kinds as follows, recalling that for the variant **Cond** *cond hint* of Definition 3, the component *cond* relates two arguments: the list of formulas it is contingent on, and the induced requirement on C and the set $S \in C$ that contained the formulas.

$\triangle$ $ps \rightsquigarrow_{\boldsymbol{\times}} qs \Longrightarrow cond\ ps\ (\lambda\text{-}\ S.\ set\ qs \cap S = \{\})$

$\triangle$ $ps \rightsquigarrow_\alpha qs \Longrightarrow cond\ ps\ (\lambda C\ S.\ set\ qs \cup S \in C)$

$\triangle$ $ps \rightsquigarrow_\beta qs \Longrightarrow cond\ ps\ (\lambda C\ S.\ \exists\, q \in set\ qs.\ \{q\} \cup S \in C)$

Conflicts are requirements only on the set S and state that none of the "consequence" formulas can be present in the set. We can express **banned** by simply putting the disallowed formula on both sides of the $\rightsquigarrow_{\boldsymbol{\times}}$ relation. The *Alpha* and *Beta* conditions look like the original ones from Definition 1 with the notable exception that we generalize to lists.

The corresponding Hintikka conditions all follow from the above when the set is maximal:

$\triangle$ $(\bigwedge ps\ qs\ q.\ ps \rightsquigarrow_{\boldsymbol{\times}} qs \Longrightarrow set\ ps \subseteq H \Longrightarrow q \in set\ qs \Longrightarrow q \notin H) \Longrightarrow hint\ H$

$\triangle$ $(\bigwedge ps\ qs\ q.\ ps \rightsquigarrow_\alpha qs \Longrightarrow set\ ps \subseteq H \Longrightarrow q \in set\ qs \Longrightarrow q \in H) \Longrightarrow hint\ H$

$\triangle$ $(\bigwedge ps\ qs.\ ps \rightsquigarrow_\beta qs \Longrightarrow set\ ps \subseteq H \Longrightarrow \exists\, q \in set\ qs.\ q \in H) \Longrightarrow hint\ H$

In Isabelle/HOL concretely, we define each kind as a locale, say *Confl*, and prove it to be a sub-locale of consistency kinds as given in Definition 7. The proofs follow Fitting's work [15]. Users instantiate the sub-locale, prove the simpler obligations, and obtain a consistency kind that they can build their consistency property with.

3.2 Gamma

Smullyan and Fitting only instantiate γ-forms with closed terms (or for term-modal logic [16] *ground* terms). In this paper, we leave the choice to the user, and work with any type of *'tm* for which the user provides a parameter substitution *map-tm*. Due to this flexibility, the user relation looks a bit different for *Gamma* than for the simpler kinds:

$\triangle$ $\rightsquigarrow_\gamma :: 'fm\ list \Rightarrow ('fm\ set \Rightarrow 'tm\ set) \times ('tm \Rightarrow 'fm\ list) \Rightarrow bool$

Here, we are asking the user to relate a list of formulas to two things: a selection function that turns a set of formulas into a set of terms, and a function that takes a term and *instantiates* the desired output with that term. The derived kind clarifies the use:

$\triangle$ $ps \rightsquigarrow_\gamma (F,\ qs) \Longrightarrow cond\ ps\ (\lambda C\ S.\ \forall\, t \in F\ S.\ set\ (qs\ t) \cup S \in C)$

We are bounding the considered terms to the output of the user-provided function on the set S. Notably, the user can provide a function that simply ignores its argument and returns whatever list of terms they want to quantify over. We use this function for two of our examples: in first-order logic, we take only the *terms* of S, proving that instantiating with terms that already occur in the derivations is sufficient for completeness; for hybrid logic, we instantiate the propositional quantifiers with formulas that satisfy a specific predicate.

The complementary Hintikka condition states that a single set contains all instances:

△ $(\bigwedge ps\ F\ qs.\ ps \rightsquigarrow_\gamma (F,\ qs) \Longrightarrow set\ ps \subseteq H \Longrightarrow (\forall\, t \in F\ H.\ set\ (qs\ t) \subseteq H)) \Longrightarrow hint\ H$

The added flexibility imposes extra requirements:

* $\bigwedge ps\ F\ qs\ f.\ ps \rightsquigarrow_\gamma (F,\ qs) \Longrightarrow (\exists\, G\ rs.\ map\ (map\text{-}fm\ f)\ ps \rightsquigarrow_\gamma (G,\ rs) \wedge$
$(\forall\, S.\ map\text{-}tm\ f\ `\ F\ S \subseteq G\ (map\text{-}fm\ f\ `\ S)) \wedge$
$(\forall\, t.\ map\ (map\text{-}fm\ f)\ (qs\ t) = rs\ (map\text{-}tm\ f\ t)))$

* $\bigwedge ps\ F\ qs\ S\ S'.\ ps \rightsquigarrow_\gamma (F,\ qs) \Longrightarrow S \subseteq S' \Longrightarrow F\ S \subseteq F\ S'$

* $\bigwedge ps\ F\ qs\ t\ A.\ ps \rightsquigarrow_\gamma (F,\ qs) \Longrightarrow t \in F\ A \Longrightarrow \exists\, B \subseteq A.\ finite\ B \wedge t \in F\ B$

The first one states that the user's relation is compatible with parameter substitutions. The middle requirement restricts the selection function to ensure it behaves well with subsets closure. The last requirement states that selected terms always arise from finite subsets: we need this when giving the family of sets finite character.

We also provide a sub-locale, *Gamma-UNIV*, that specializes the *Gamma* locale to the full universe of terms. This simplifies the cases where the extra flexibility clutters the definitions.

3.3 Delta

The *Delta* locale is perhaps the simplest of them all, as we simply expose the **Wits** variant from the base locale (Definition 3). We only ask for a witnessing function that respects parameter substitutions:

△ $\delta :: 'fm \Rightarrow 'x \Rightarrow 'fm\ list$

* $\bigwedge p\ f\ x.\ \delta\ (map\text{-}fm\ f\ p)\ (f\ x) = map\ (map\text{-}fm\ f)\ (\delta\ p\ x)$

3.4 Modal

Our final specialization comes from Fitting's paper on term-modal logic [16]. It remains future work to use it for a concrete logic and proof system, but we have proved that the locale constitutes a consistency kind as defined in Definition 7.

As the Hintikka property depends on the concrete logic, we make our locale parametric over both that and the relation on formulas. The obligations are similar to those for *Gamma*.

△ $\rightsquigarrow_\Box :: 'fm\ list \Rightarrow ('fm\ set \Rightarrow 'fm\ set) \times 'fm\ list \Rightarrow bool$

△ $hint :: 'fm\ set \Rightarrow bool$

△ $ps \rightsquigarrow_\Box (F,\ qs) \Longrightarrow cond\ ps\ (\lambda C\ S.\ set\ qs \cup F\ S \in C)$

3.5 Discussion

We have chosen to let the user specify the Hintikka condition associated with their consistency kind. Looking at the ones above, however, we see that they often closely resemble each other. If we had given a more restricted definition of abstract kinds, we might have been able to derive the Hintikka counterpart automatically. However, by using sub-locales, we have provided an interface where they are effectively pre-defined. As such, we have opted for the extra degree of freedom in our base locales. We have also made use of a compositionality that Smullyan and Fitting never seem to spell out explicitly: these kinds are all independent of each other. We do not have to fix a consistency property in advance, as in Definition 1, and provide one big, multi-case proof that we can impose finite character. Instead, we can prove the required properties of each kind separately and assemble the whole afterwards.

4 Derivational Consistency

We set out to prove completeness by falsifying underivable formulas. The abstract consistency properties split this process in two: (1) build a model from a maximal consistent set and (2) prove that underivability is an abstract consistency property. So far, we have focused on part one, but our development also provides machinery for part two: the *derivational kinds*.

▶ **Definition 14** (Derivational Kinds). *A consistency kind K (cf. Definition 7) is a* derivational kind *with respect to a derivational consistency predicate $\vdash$ - when K is satisfied on the family of consistent sets that leave enough parameters new (given infinitely many formulas):*

△ $\vdash$ - :: *'fm set* ⇒ *bool*

∗ *infinite* (*UNIV* :: *'fm set*) ⟹ sat_E K {*A*. *enough-new A* ∧ $\vdash$ *A*}

A list of derivational kinds constitutes a consistency property, paving the way for completeness. All the sub-locales we presented above can be shown to be derivational kinds, subject to a few requirements on the user's calculus:

∗ $\bigwedge S\ ps\ qs\ x.\ set\ ps \subseteq S \Longrightarrow ps \rightsquigarrow\!\!\!\!\times\ qs \Longrightarrow x \in set\ qs \Longrightarrow x \in S \Longrightarrow \neg \vdash S$

∗ $\bigwedge S\ ps\ qs.\ set\ ps \subseteq S \Longrightarrow ps \rightsquigarrow_\alpha qs \Longrightarrow \vdash S \Longrightarrow \vdash set\ qs \cup S$

∗ $\bigwedge S\ ps\ qs.\ set\ ps \subseteq S \Longrightarrow ps \rightsquigarrow_\beta qs \Longrightarrow \vdash S \Longrightarrow \exists q \in set\ qs.\ \vdash \{q\} \cup S$

∗ $\bigwedge S\ ps\ F\ qs\ t.\ set\ ps \subseteq S \Longrightarrow ps \rightsquigarrow_\gamma (F,\ qs) \Longrightarrow t \in F\ S \Longrightarrow \vdash S \Longrightarrow \vdash set\ (qs\ t) \cup S$

∗ $\bigwedge S\ p\ x.\ p \in S \Longrightarrow x \notin params\ S \Longrightarrow \vdash S \Longrightarrow \vdash set\ (\delta\ p\ x) \cup S$

∗ $\bigwedge S\ ps\ F\ qs.\ set\ ps \subseteq S \Longrightarrow ps \rightsquigarrow_\Box (F,\ qs) \Longrightarrow \vdash S \Longrightarrow \vdash set\ qs \cup F\ S$

For *Confl*, any set that contains conflicting formulas is inconsistent. For *Alpha*, the sub-forms together preserve consistency. For *Beta*, at least one sub-form preserves consistency. For *Gamma*, any instance preserves consistency. For *Delta*, any instance from a new parameter preserves consistency. For *Modal*, the transformation preserves consistency.

▶ Remark 15. The above proof obligations arise from unfolding and simplifying definitions. They are what satisfaction of the various kinds specialize to on the family of derivationally consistent sets. However, we find that simplifying them on the library side makes Isabelle/HOL's automation work better on the user side: in many cases the "sledgehammer" tool [5] can piece together the concrete proof rules needed to prove the obligation.

5 Completeness Proofs for Three Logics

In this section we apply our development to three case studies: first-order logic with a restricted instantiation rule, second-order logic, and Prior's Ideal Language. For first-order logic we have proved compactness, and strong completeness for natural deduction and tableau systems. For second-order logic, we have proved weak completeness of an axiomatic system and strong completeness for natural deduction. Finally, for hybrid logic we have proved strong completeness for a natural deduction system. We cover the most pertinent aspects here, and refer to the supplementary material for the full details.

5.1 Bounded First-Order Logic

We call our first-order logic "bounded" in the sense of From and Jacobsen [22]: the instantiation rule for the universal quantifier only applies to already occurring terms, and we build the model over this *bounded* domain of terms.

△ $[\ \bot\ ] \rightsquigarrow\!\!\!\!\times\ [\ \bot\ ]$
△ $[\ \neg\ (\cdot P\ ts)\ ] \rightsquigarrow\!\!\!\!\times\ [\ \cdot P\ ts\ ]$
△ $[\ \neg\ (p \longrightarrow q)\ ] \rightsquigarrow_\alpha [\ p, \neg\ q\ ]$
△ $[\ p \longrightarrow q\ ] \rightsquigarrow_\beta [\ \neg\ p, q\ ]$
△ $[\ \forall p\ ] \rightsquigarrow_\gamma (terms, \lambda t.\ [\ \langle t \rangle p\ ])$
△ $\delta\ (\neg\ \forall p)\ x = [\ \neg\ \langle \star x \rangle p\ ]$

Figure 1 Bounded first-order logic consistency property.

△ $p \in A \Longrightarrow A \Vdash p$
△ $A \Vdash \bot \Longrightarrow A \Vdash p$
△ $\{p\} \cup A \Vdash q \Longrightarrow A \Vdash (p \longrightarrow q)$
△ $A \Vdash (p \longrightarrow q) \Longrightarrow A \Vdash p \Longrightarrow A \Vdash q$

△ $A \Vdash \langle \star a \rangle p \Longrightarrow a \notin P.params\ (\{p\} \cup A) \Longrightarrow A \Vdash \forall p$
△ $A \Vdash \forall p \Longrightarrow t \in terms\ (\{p\} \cup A) \Longrightarrow A \Vdash \langle t \rangle p$
△ $\{p \longrightarrow q\} \cup A \Vdash p \Longrightarrow A \Vdash p$

Figure 2 Bounded natural deduction for first-order logic.

We evaluate our first-order logic on models *Model U E F G* consisting of a domain $U :: 'a\ set$, represented explicitly as a set, an environment $E :: nat \Rightarrow 'a$ that maps (de Bruijn) variables to elements of the domain, a function denotation $F :: 'f \Rightarrow 'a\ list \Rightarrow 'a$ and a predicate denotation $G :: 'p \Rightarrow 'a\ list \Rightarrow bool$. We only consider well formed models where the environment and function denotations respect the domain:
△ *wf-model* $(Model\ U\ E\ F\ G) \longleftrightarrow (\forall n.\ E\ n \in U) \wedge (\forall f\ ts.\ F\ f\ ts \in U)$

Higher-order logic makes the semantics straightforward to express. Note that bolded connectives, e.g. $\longrightarrow$, belong to the embedded logic (here first-order logic).
△ $(\text{-} \models \bot) = False$
△ $(Model\ \text{-}\ E\ F\ G \models \cdot P\ ts) = G\ P\ (map\ \langle\!|(E, F)|\!\rangle\ ts)$
△ $(Model\ U\ E\ F\ G \models p \longrightarrow q) = (Model\ U\ E\ F\ G \models p \longrightarrow Model\ U\ E\ F\ G \models q)$
△ $(Model\ U\ E\ F\ G \models \forall p) = (\forall x \in U.\ Model\ U\ (x \gg E)\ F\ G \models p)$

The map $\langle\!|\text{-}|\!\rangle$ evaluates terms to elements of the domain. Terms are either variables or function symbols applied to a list of terms. The operation $x \gg E$ shifts environment E by mapping variable 0 to x and every other variable $n + 1$ to $E\ n$.

From these semantics, we write out the abstract consistency property in Figure 1 using the pre-defined kinds. Falsity conflicts with itself and negated predicates conflict with their non-negated counterparts. Negated implication has an α-form, while implication itself has β-form. For the positive quantifier, we only instantiate with the *terms* of the given set, and for the negative quantifier we simply instantiate with a constant (denoted by the star). The declarations in Figure 1 are valid Isabelle/HOL code, and, in our opinion, almost as readable as the textbook declarations that we started from in Definition 1.

We use a Herbrand model with a small twist as our canonical model:
△ *canonical* $H \equiv$
$Model\ (terms\ H)\ (\lambda n.\ \#n \in ?\ terms\ H)\ (\lambda f\ ts.\ \bigcirc f\ ts \in ?\ terms\ H)\ (\lambda P\ ts.\ \cdot P\ ts \in H)$

Here, $\#$ and $\bigcirc$ form variables and composed terms, respectively. The domain consists only of the terms of the given (maximal consistent) set H. To guarantee well formedness, we *guard* the denotations with $\in ?$, which returns the left-hand side if the right-hand side contains it, and an arbitrary element of the right-hand side otherwise. Assuming at least one term, we prove a truth lemma for the canonical model by induction on the size of the formula, using the Hintikka properties of the set to discharge each case.

Figure 2 shows a natural deduction system for first-order logic with two notable details: we are using sets of formulas, and we can only eliminate the universal quantifier with a term

$\triangle$ $[\,\forall\, p\,] \rightsquigarrow_{\gamma} (\lambda t.\ [\,\langle t/0\rangle p\,])$
$\triangle$ $[\,\forall_F\ p\,] \rightsquigarrow_{\gamma F} (\lambda s.\ [\,\langle s/0\rangle_F\ p\,])$
$\triangle$ $[\,\forall_P\ p\,] \rightsquigarrow_{\gamma P} (\lambda s.\ [\,\langle s/0\rangle_P\ p\,])$

$\triangle$ $\delta\ (\neg\ \forall\, p)\ x = [\,\neg\ \langle \star x/0\rangle\ p\,]$
$\triangle$ $\delta\ (\neg\ \forall_P\ p)\ x = [\,\neg\ \langle \bigcirc_2\ x/0\rangle_P\ p\,]$
$\triangle$ $\delta\ (\ \neg\ \forall_F\ p\ )\ x = [\,\neg\ \langle \bigcirc_2\ x/0\rangle_F\ p\,]$

Figure 3 Second-order logic consistency property (partial).

from the derivation. Proving completeness directly for this restricted calculus is a stronger result than typically found. We prove soundness over well formed models by rule induction. For strong completeness, we show that the sets from which we *cannot* derive falsity (and which leave enough parameters new) form a consistency property. Concretely we instantiate the derivational locales, benefitting from the close correspondence between Figures 1 and 2.

▶ **Theorem 16** (FOL Completeness). *If the set of formulas A leaves enough parameters new, and all well formed models that satisfy A also satisfy formula p, then we can derive p from A:*

- $\bigwedge(U :: 'f\ tm\ set)\ E\ F\ G.\ wf\text{-}model\ (Model\ U\ E\ F\ G) \Longrightarrow$
  $(\forall\, q \in A.\ Model\ U\ E\ F\ G \models q) \Longrightarrow Model\ U\ E\ F\ G \models p$
- $P.enough\text{-}new\ A$

$\therefore$ $A \Vdash p$

If we prefer lists of assumptions to sets, we can easily modify the proof system in Figure 2 to use lists instead. Strong completeness of the resulting system (if p follows from a set A then we can derive p from a list of elements from A) follows by noticing that each proof rule in Figure 2 relies only on a *finite kernel* of formulas. Rule induction lets us go from a set derivation to derivation from a list of formulas, reusing the completeness result.

The supplementary material contains strong completeness for a tableau system whose elimination rule is limited to the already occurring terms. Since we can reuse model existence, we can define this system and prove its soundness and completeness in less than 150 lines of Isabelle/HOL. Reusing model existence to prove compactness takes around 200 lines. The first-order logic example in total consists of around 1000 lines.

5.2 Second-Order Logic

We use the framework to obtain the, to our knowledge, first formalization in Isabelle/HOL of second-order logic and a natural deduction system for it. We also use the framework to obtain the, to our knowledge, first formalization in a proof assistant of an axiomatic system for second-order logic. The formalization of the syntax and semantics of second-order logic generalizes a formalization of first-order logic by From [18, 23]. The generalization consists of three steps. (1) defining predicate and function symbols as being either proper symbols or variables. (2) Using these symbols in the definition of predicate and function applications. (3) Introducing two new quantifiers – one, $\forall_F$, that binds variables to functions and another, $\forall_P$, that binds variables to predicates. Second-order logic is famously incomplete for standard models [32] where the quantifiers range over all functions and predicates over the universe. However completeness holds when considering general models, i.e., models that include the sets of predicates and functions over which the quantifiers can range. We formalize the latter. Note that we do not consider *frugal* models [1]; we do not know the sizes of the models.

△ $[\ (i, \neg\ \cdot P)\ ] \rightsquigarrow_{\boldsymbol{\times}} [(i, \cdot P)]$
△ $[\ (i, \neg\ \bullet k)\ ] \rightsquigarrow_{\boldsymbol{\times}} [(i, \bullet k)]$
△ $[\ (i, p \wedge q)\ ] \rightsquigarrow_{\alpha} [(i, p), (i, q)]$
△ $[\ (i, \neg\ (p \wedge q))\ ] \rightsquigarrow_{\beta} [(i, \neg\ p), (i, \neg\ q)]$
△ $[\ (i, \neg\ \neg\ p)\ ] \rightsquigarrow_{\alpha} [(i, p)]$

△ $[\ (i, @k\ p)\ ] \rightsquigarrow_{\alpha} [(k, p)]$
△ $[\ (i, \neg\ @k\ p)\ ] \rightsquigarrow_{\alpha} [(k, \neg\ p)]$
△ $[\ (i, \downarrow p)\ ] \rightsquigarrow_{\alpha} [\ (i, \langle i \rangle_i\ p)\ ]$
△ $[\ (i, \neg \downarrow p)\ ] \rightsquigarrow_{\alpha} [\ (i, \neg\ \langle i \rangle_i\ p)\ ]$
△ $[\ (i, \Box p), (i, \Diamond(\bullet k))\ ] \rightsquigarrow_{\alpha} [(k, p)]$
△ $[\ (i, \mathbf{A}\ p)\ ] \rightsquigarrow_{\gamma i} (\lambda k.\ [\ (k, p)\ ])$

△ $[\ (i, \forall\ p)\ ] \rightsquigarrow_{\gamma p} (\lambda\text{-}.\ \{q.\ \mathit{softqdf}\ q\}, \lambda q.\ [\ (i, \langle q \rangle_p\ p)\ ])$

△ $[] \rightsquigarrow_{\gamma i} (\lambda i.\ [\ (i, \bullet i)\ ])$
△ $[\ (i, \bullet k)\ ] \rightsquigarrow_{\alpha} [(k, \bullet i)]$
△ $[\ (i, \bullet k), (i, p)\ ] \rightsquigarrow_{\alpha} [(k, p)]$

△ $\delta\ (i, \neg\ \Box p)\ k = [(\bigcirc k, \neg\ p), (i, \Diamond\ (\bullet\ (\bigcirc\ k)))]$
△ $\delta\ (i, \neg\ \mathbf{A}\ p)\ k = [\ (\bigcirc k, \neg\ p)\ ]$
△ $\delta\ (i, \neg\ \forall\ p)\ P = [\ (i, \neg\ \langle \cdot(\bigcirc P) \rangle_p\ p)\ ]$

Figure 4 Consistency property for Prior's Ideal Language.

A general model for our second-order logic is a tuple $(E, E_F, E_P, C, F, G, PS, FS)$, where $E :: nat \Rightarrow 'a$ is the environment for variables over elements, $'a$ is the type representing the domain or universe, $E_F :: nat \Rightarrow ('a\ list \Rightarrow 'a)$ is the environment for function variables, $E_P :: nat \Rightarrow ('a\ list \Rightarrow bool)$ is the environment for predicate variables, $C :: 'b \Rightarrow 'a$ is the constant denotation, $F :: 'b \Rightarrow ('a\ list \Rightarrow 'a)$ is the function denotation, $G :: 'b \Rightarrow ('a\ list \Rightarrow bool)$ is the predicate denotation, $PS :: ('a\ list \Rightarrow bool)\ set$ is the predicate universe and $FS :: ('a\ list \Rightarrow bool)\ set$ is the function universe. A model is well formed when its predicate/function environments and denotations stay within the defined universes:

△ *wf-model* $(E,\ E_F,\ E_P,\ C,\ F,\ G,\ PS,\ FS) \longleftrightarrow$
$\quad range\ G \subseteq PS \wedge range\ E_P \subseteq PS \wedge range\ F \subseteq FS \wedge range\ E_F \subseteq FS$

Since the universe of elements is represented by the type $'a$, the typing of E and C already ensures that these stay within the universe.

The conflict, alpha, and beta kinds are the same as for bounded first-order logic in Figure 1. Figure 3 defines gamma and delta rules for second-order logic that correspond to the three universal quantifiers of the language. Notably, we can use the framework with both single-point substitution (here) and simultaneous substitution (for FOL and PIL). We again build a Herbrand model over the resulting maximal consistent set. We do not care about distinguishing open and closed formulas, though such a distinction could easily be made by tweaking the consistency property and canonical model. Model existence follows almost immediately from the Hintikka properties.

Our axiomatic system for second-order logic generalizes From's work [18, 23] by introducing axioms and rules for the second-order quantifiers. The rules are generalization rules for introducing universal quantifications over predicates and functions, respectively. The axioms are instantiation axioms stating that universal quantifications, over respectively predicates and functions, imply their immediate sub-formula with the quantified variable substituted with any symbol. Our natural deduction system is a variant of the natural deduction system for the first-order logic above without the elimination limitation, but with introduction and elimination rules for the second-order quantifiers. We prove weak completeness for the axiomatic system and strong completeness for the natural deduction system. The entire development consists of under 1200 lines.

5.3 Prior's Ideal Language

We have mechanized the completeness of a recently developed [4] strong hybrid logic with propositional quantification. The combination of Kripke semantics and propositional quantification gives some second-order expressibility, resulting in a system of significant strength. The mechanization of this example consists of just under 1800 lines.

We interpret the logic on Kripke models with explicit sets of admissible propositions. This set must be closed under relative complement, finite intersection and modal projection, corresponding to negation, conjunction and the box modality. The three most interesting cases for the semantics are the global modality, downarrow binder, and propositional quantification:

$\triangle$ $((M, \text{-}) \models \mathbf{A}\ p) = (\forall\, v \in \mathcal{W}\ M.\ (M, v) \models p)$

$\triangle$ $((M, w) \models \downarrow p) = ((M(\!|\mathfrak{N} := (w \gg \mathfrak{N}\ M)|\!), w) \models p)$

$\triangle$ $((M, w) \models \forall\ p) = (\forall\, P \in \Pi\ M.\ (M(\!|\mathfrak{V} := (P \gg \mathfrak{V}\ M)|\!), w) \models p)$

Figure 4 demonstrates the flexibility of our abstract consistency properties. Our formulas are labelled by *nominals* i, k, a special sort of propositional symbol. A nominal is true at exactly one world in a frame, allowing it to *name* that world. The first two lines of the consistency property state conflicts for propositions and nominals, respectively. The next three lines characterize the conjunction and negation operators.

The satisfaction operator, @, shifts evaluation to the world named by the given nominal. We represent this by changing the label with an alpha kind. Likewise, the downarrow binder, $\downarrow$, names the current point of evaluation, and again alpha kinds suffice: an instance at the current label gives evidence for the downarrow. For the box modality, $\Box$, we take advantage of the ability to condition on multiple formulas. Naively, we could characterize the box modality with a gamma property, but this modality is decidedly *local*: it talks only about worlds we can *see* (here k), so alpha suffices. We define the diamond modality as usual: $\Diamond\ p \equiv \neg\ (\Box\ (\neg\ p))$. The global modality, $\mathbf{A}$, on the other hand, is decidedly global, and can only be captured by a gamma property. Here we instantiate the *label* rather than the formula itself, making full use of our flexible setup.

The middle gamma property instantiates the propositional quantifier with every *softqdf* formula: the quantifier- and downarrow-free formulas with no nominals in formula position [4]. Intuitively, these are the formulas that "stay inside" the admissible propositions and the mechanization helped clarify that the downarrow binder should be excluded from this class.

The next three lines state that our nominals are reflexive, symmetric, and satisfy the same formulas. Notice how we use an unconditional gamma kind to characterize reflexivity. The three delta kinds are straightforward. The first one again demonstrates the locality of the box modality: we can *see* the world where the formula does not hold.

We build the canonical model in the style of Braüner [10] by using representatives of equivalence classes of nominals as our worlds. Otherwise, we use the original paper's model construction [4]. Notably, however, we prove strong completeness of a natural deduction system instead of the original axiomatic system. Moreover, the original paper tweaks the Lindenbaum construction itself to ensure that the maximal consistent set has the properties they need for their model. In contrast, we get the (Hintikka) properties we need from the abstract consistency property, which, in a sense, acts as an interface to control the Lindenbaum construction. We have proved the details of the Lindenbaum lemma for any logic where our abstract consistency properties can express what we need, and for these examples they can. Our proof system follows the consistency property in Figure 4 closely.

6 Related Work

We are not the first to mechanize parts of Fitting's textbook [15] in Isabelle/HOL. Berghofer [3] does not make any of the generalizations we do, but mechanizes the (weak) completeness of a natural deduction system and downward Löwenheim-Skolem results for (countable) first-order logic. Michaelis and Nipkow [30, 31] make greater use of the applicability of the model existence theorem and prove completeness for a range of proof systems in their propositional development. They prove compactness, cut elimination and interpolation results. Suaréz et al. [40, 37] have also mechanized propositional compactness using the same approach as us and used it for combinatorial applications. Doty [13], on the other hand, used Zorn's lemma to construct their maximal consistent sets for any cardinality in their abstract mechanization of classical propositional logic, including strong soundness and completeness. Harrison [24] used HOL Light to mechanize compactness and the Löwenheim-Skolem theorem for first-order logic. He used a similar method to extend a consistent set into a maximal one as we do, but chose to assume a countable language. Tourret and Paulson [41] made the same choice in their recent translation of Harrison's work to Isabelle/HOL. Schlöder and Koepke [36] notably mechanized Gödel's completeness theorem for uncountable languages in Mizar.

Blanchette et al. [8] used coinductive methods to develop a more operational Isabelle/HOL framework for completeness proofs than ours. Where we seek to characterize maximal consistent sets, they provide machinery for building derivation trees and constructing a model out of those. This gives a direct connection to executable provers, and several entries in the Isabelle Archive of Formal Proofs [11, 22, 19, 21] make use of this. From [20] recently developed an Isabelle/HOL framework for *synthetic* completeness proofs (not to be confused with the synthetic style of metatheory [26, 27] used in Rocq developments). That work also mechanizes a construction of maximal consistent sets, but limits itself to a calculus-specific notion of consistency and relies heavily on cut rules. Our flexibility allows for a wider range of applications, including completeness proofs for analytic calculi like tableau, and proofs of compactness, interpolation, etc. For their example proof systems, From [20] either includes rules for weakening and cut or shows them to be admissible before being able to apply the framework. We do not need to do so in our setting. Petria's [34] "institutions" give a category-theoretic alternative to abstract consistency properties for proving a general completeness theorem, but this work has not yet been mechanized in a proof assistant.

Koch et al. [28, 29] mechanized undecidability, incompleteness, and completeness results for second-order logic in Rocq. Their work significantly exceeds our second-order logic completeness example. For one thing, they consider Henkin models that satisfy the comprehension axiom, whereas we only work with the cruder notion of general models. Our example demonstrates that abstract consistency properties scale to second-order logic, but our examples does not go beyond that. We do, however, believe it to be the first such result in Isabelle/HOL specifically. Our mechanization of Prior's Ideal Language [4] is certainly the first of its kind, though completeness results for weaker hybrid logics do exist [17, 20]. Considering general models for higher-order logic itself, Díaz [12] and Schlichtkrull [35] have both mechanized the soundness of Andrews' Q_0 system [1], a formulation of Church's Simple Theory of Types.

7 Discussion

We have mechanized a very general formulation of Smullyan and Fitting's abstract consistency properties and used locales to recover the ease-of-use of the original version. Our three case studies demonstrate the applicability of our development: what started as a unifying

principle for first-order logic [38] proves to be just as useful for second-order and hybrid logic. Our compositional, abstract consistency properties provide a declarative interface for characterizing maximal consistent sets and free us from tinkering with the details of the Lindenbaum lemma regardless of how many quantifiers we have, or what we consider to be the "formulas" and "terms" of our logic. As such, we hope that these consistency properties can be a unifying principle for more than just first-order logic.

To this effect, we want to develop our theory further, and especially find its limits. We noted Smullyan and Fitting's many applications, but have so far only mechanized completeness and compactness. The rest remain future work. We have scaled to second-order logic for general models, but we want to develop the example further by including the comprehension axiom and considering Henkin models. We are also curious about higher-order logic itself. Our gamma kinds apply to first-order logic terms, but also scale to hybrid logic formulas, so maybe we can use lambda terms as well. Finally, we want to consider intuitionistic logics. Kripke semantics for modal logic are obviously not an obstacle, but we would like to take it further. We have built the consistency properties in this paper in the same style as Smullyan, by considering formulas and their negations, but our generalization to lists might allow us to be more ecumenical. Instead of characterizing implication by its classical interpretation as a disjunction, we could perhaps consider a list of $\phi \to \psi$ and ϕ as an α-form with sub-form ψ. Our setup is certainly flexible enough, but we are yet to explore the consequences.

Looking at our existing examples, we are also wondering whether we can gainfully characterize the connection between semantics and consistency property. Perhaps we can apply some of our machinery to soundness proofs as well as completeness proofs. On the more mundane scale, we are also interested in mechanizing resolution and sequent calculus.

We have exploited a compositionality of abstract consistency properties that we have not seen Smullyan or Fitting spell out explicitly. This compositionality proves very useful when we want to characterize logics with different kinds of γ-forms. If we tried to characterize all these forms with the same *Gamma* kind, we would have to quantify over a universe containing all the types of terms we are interested in. Instead, we compose separate *Gamma* kinds, each with their own type of terms.

We are, however, somewhat unsatisfied with using a list of kinds to define our composition, and wonder if there is a better way to do it. We define consistency kinds as a locale, a very meta-logical mechanism, compose them in a list, a much more object-level notion, and then try to work with the list inside another locale. We found that this stifles Isabelle/HOL automation, and that to achieve reasonable performance, we need to specify, on the user side, exactly which kind in our list we currently need. We wonder if there is a way to compose a variadic number of kinds whilst staying in the meta-logical language of locales. Such compositionality might have other uses than ours. Alternatively, we would be interested in machinery for automatically specializing our lemmas to each element of the user-given list.

References

1 Peter B. Andrews. *An introduction to mathematical logic and type theory - to truth through proof*. Computer science and applied mathematics. Academic Press, 1986.

2 Clemens Ballarin. Locales: A module system for mathematical theories. *Journal of Automated Reasoning*, 52(2):123–153, 2014. doi:10.1007/S10817-013-9284-7.

3 Stefan Berghofer. First-order logic according to Fitting. *Archive of Formal Proofs*, 2007, August 2007. Formal proof development. URL: https://www.isa-afp.org/entries/FOL-Fitting.shtml.

4 Patrick Blackburn, Torben Braüner, and Julie Lundbak Kofod. Prior's ideal language. *Mathematical Structures in Computer Science*, 35, 2025. doi:10.1017/S0960129525000076.

5 Jasmin Christian Blanchette, Sascha Böhme, and Lawrence C. Paulson. Extending Sledgehammer with SMT solvers. *Journal of Automated Reasoning*, 51(1):109–128, 2013. doi:10.1007/S10817-013-9278-5.

6 Jasmin Christian Blanchette, Johannes Hölzl, Andreas Lochbihler, Lorenz Panny, Andrei Popescu, and Dmitriy Traytel. Truly modular (co)datatypes for Isabelle/HOL. In Gerwin Klein and Ruben Gamboa, editors, *Interactive Theorem Proving - 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8558 of *Lecture Notes in Computer Science*, pages 93–110. Springer, 2014. doi:10.1007/978-3-319-08970-6_7.

7 Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel. Cardinals in Isabelle/HOL. In Gerwin Klein and Ruben Gamboa, editors, *Interactive Theorem Proving - 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8558 of *Lecture Notes in Computer Science*, pages 111–127. Springer, 2014. doi:10.1007/978-3-319-08970-6_8.

8 Jasmin Christian Blanchette, Andrei Popescu, and Dmitriy Traytel. Soundness and completeness proofs by coinductive methods. *Journal of Automated Reasoning*, 58(1):149–179, 2017. doi:10.1007/s10817-016-9391-3.

9 Anthony Bordg, Lawrence C. Paulson, and Wenda Li. Simple Type Theory is not too simple: Grothendieck's schemes without dependent types. *Exp. Math.*, 31(2):364–382, 2022. doi:10.1080/10586458.2022.2062073.

10 Torben Braüner. *Hybrid Logic and its Proof-Theory*. Springer, 1st edition, 2011. doi:10.1007/978-94-007-0002-4.

11 Joachim Breitner and Denis Lohner. The meta theory of the Incredible Proof Machine. *Archive of Formal Proofs*, 2016, May 2016. Formal proof development. URL: https://www.isa-afp.org/entries/Incredible_Proof_Machine.shtml.

12 Javier Díaz. Metatheory of Q0. *Archive of Formal Proofs*, 2023, November 2023. Formal proof development. URL: https://www.isa-afp.org/entries/Q0_Metatheory.html.

13 Matthew Doty. Class-based classical propositional logic. *Archive of Formal Proofs*, 2022, December 2022. Formal proof development. URL: https://www.isa-afp.org/entries/Propositional_Logic_Class.html.

14 Melvin Fitting. Model existence theorems for modal and intuitionistic logics. *Journal of Symbolic Logic*, 38(4):613–627, 1973. URL: 10.2307/2271986, doi:10.2307/2271986.

15 Melvin Fitting. *First-Order Logic and Automated Theorem Proving, Second Edition*. Graduate Texts in Computer Science. Springer, 1996. doi:10.1007/978-1-4612-2360-3.

16 Melvin Fitting, Lars Thalmann, and Andrei Voronkov. Term-modal logics. *Studia Logica*, 69(1):133–169, 2001. doi:10.1023/A:1013842612702.

17 Asta Halkjær From. Synthetic completeness for a terminating Seligman-style tableau system. In Ugo de'Liguoro, Stefano Berardi, and Thorsten Altenkirch, editors, *26th International Conference on Types for Proofs and Programs, TYPES 2020, March 2-5, 2020, University of Turin, Italy*, volume 188 of *LIPIcs*, pages 5:1–5:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.TYPES.2020.5.

18 Asta Halkjær From. Soundness and completeness of an axiomatic system for first-order logic. *Archive of Formal Proofs*, 2021, September 2021. Formal proof development. URL: https://www.isa-afp.org/entries/FOL_Axiomatic.html.

19 Asta Halkjær From. A naive prover for first-order logic. *Archive of Formal Proofs*, 2022, March 2022. Formal proof development. URL: https://www.isa-afp.org/entries/FOL_Seq_Calc3.html.

20 Asta Halkjær From. An Isabelle/HOL framework for synthetic completeness proofs. In Kathrin Stark, Amin Timany, Sandrine Blazy, and Nicolas Tabareau, editors, *Proceedings of the 14th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP*

2025, Denver, CO, USA, January 20-21, 2025, CPP '25, pages 171–186, New York, NY, USA, 2025. ACM. doi:10.1145/3703595.3705882.

21 Asta Halkjær From and Frederik Krogsdal Jacobsen. A sequent calculus prover for first-order logic with functions. *Archive of Formal Proofs*, 2022, January 2022. Formal proof development. URL: https://www.isa-afp.org/entries/FOL_Seq_Calc2.html.

22 Asta Halkjær From and Frederik Krogsdal Jacobsen. Verifying a sequent calculus prover for first-order logic with functions in Isabelle/HOL. *Journal of Automated Reasoning*, 68(3):15, 2024. doi:10.1007/S10817-024-09697-3.

23 Asta Halkjær From. A succinct formalization of the completeness of first-order logic. In Henning Basold, Jesper Cockx, and Silvia Ghilezan, editors, *27th International Conference on Types for Proofs and Programs, TYPES 2021, June 14-18, 2021, Leiden, The Netherlands (Virtual Conference)*, volume 239 of *LIPIcs*, pages 8:1–8:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.TYPES.2021.8.

24 John Harrison. Formalizing basic first order model theory. In Jim Grundy and Malcolm C. Newey, editors, *Theorem Proving in Higher Order Logics, 11th International Conference, TPHOLs'98, Canberra, Australia, September 27 - October 1, 1998, Proceedings*, volume 1479 of *Lecture Notes in Computer Science*, pages 153–170. Springer, 1998. doi:10.1007/BFb0055135.

25 Leon Henkin. The discovery of my completeness proofs. *Bulletin of Symbolic Logic*, 2(2):127–158, 1996. doi:10.2307/421107.

26 Dominik Kirst and Marc Hermes. Synthetic undecidability and incompleteness of first-order axiom systems in Coq. *Journal of Automated Reasoning*, 67(1):13, 2023. doi:10.1007/S10817-022-09647-X.

27 Dominik Kirst and Benjamin Peters. Gödel's theorem without tears - essential incompleteness in synthetic computability. In Bartek Klin and Elaine Pimentel, editors, *31st EACSL Annual Conference on Computer Science Logic, CSL 2023, February 13-16, 2023, Warsaw, Poland*, volume 252 of *LIPIcs*, pages 30:1–30:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.CSL.2023.30.

28 Mark Koch. Mechanizing second-order logic in Coq. Bachelor's thesis, Saarland University, 2021.

29 Mark Koch and Dominik Kirst. Undecidability, incompleteness, and completeness of second-order logic in Coq. In Andrei Popescu and Steve Zdancewic, editors, *CPP '22: 11th ACM SIGPLAN International Conference on Certified Programs and Proofs, Philadelphia, PA, USA, January 17 - 18, 2022*, pages 274–290. ACM, 2022. doi:10.1145/3497775.3503684.

30 Julius Michaelis and Tobias Nipkow. Propositional proof systems. *Archive of Formal Proofs*, 2017, June 2017. Formal proof development. URL: https://www.isa-afp.org/entries/Propositional_Proof_Systems.shtml.

31 Julius Michaelis and Tobias Nipkow. Formalized proof systems for propositional logic. In Andreas Abel, Fredrik Nordvall Forsberg, and Ambrus Kaposi, editors, *23rd International Conference on Types for Proofs and Programs (TYPES 2017)*, volume 104 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1–16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. Paper 5. doi:10.4230/LIPIcs.TYPES.2017.5.

32 Richard Montague. Reductions of higher-order logic. In J.W. Addison, Leon Henkin, and Alfred Tarski, editors, *The Theory of Models: Proceedings of the 1963 International Symposium at Berkeley*, Studies in Logic and the Foundations of Mathematics, pages 251–264. North-Holland, 1963. doi:10.1016/B978-0-7204-2233-7.50030-7.

33 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002. URL: https://isabelle.in.tum.de/doc/tutorial.pdf, doi:10.1007/3-540-45949-9.

34 Marius Petria. An institutional version of Gödel's completeness theorem. In Till Mossakowski, Ugo Montanari, and Magne Haveraaen, editors, *Algebra and Coalgebra in Computer Science, Second International Conference, CALCO 2007, Bergen, Norway, August 20-24, 2007, Pro-*

ceedings, volume 4624 of *Lecture Notes in Computer Science*, pages 409–424. Springer, 2007. doi:10.1007/978-3-540-73859-6_28.

35 Anders Schlichtkrull. Soundness of the Q0 proof system for higher-order logic. *Archive of Formal Proofs*, 2023, November 2023. Formal proof development. URL: https://www.isa-afp.org/entries/Q0_Soundness.html.

36 Julian J. Schlöder and Peter Koepke. The Gödel completeness theorem for uncountable languages. *Formalized Mathematics*, 20(3):199–203, 2012. doi:10.2478/v10037-012-0023-z.

37 Fabián Fernando Serrano Suárez, Mauricio Ayala-Rincón, and Thaynara Arielly de Lima. Hall's theorem for enumerable families of finite sets. In Kevin Buzzard and Temur Kutsia, editors, *Intelligent Computer Mathematics*, pages 107–121, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-16681-5_7.

38 Raymond M. Smullyan. A unifying principal in quantification theory. *Proceedings of the National Academy of Sciences*, 49(6):828–832, 1963. doi:10.1073/pnas.49.6.828.

39 Raymond M. Smullyan. *First-Order Logic*. Springer, Berlin, 1968. doi:10.1007/978-3-642-86718-7.

40 Fabián Fernando Serrano Suárez, Thaynara Arielly de Lima, and Mauricio Ayala-Rincón. Compactness theorem for propositional logic and combinatorial applications. *Archive of Formal Proofs*, 2024, August 2024. Formal proof development. URL: https://www.isa-afp.org/entries/Prop_Compactness.html.

41 Sophie Tourret and Lawrence C. Paulson. Compactness theorem for first-order logic. *Archive of Formal Proofs*, 2025, February 2025. Formal proof development. URL: https://www.isa-afp.org/entries/FOL_Compactness.html.