Henkin and Hybrid Logic[*]

Patrick Blackburn
Roskilde Universitet

Antonia Huertas
Universitat Oberta de Catalunya

María Manzano
Universidad de Salamanca

Klaus Frovin Jørgensen
Roskilde Universitet

July 10, 2014

Abstract

Leon Henkin was not a modal logician, but there is a branch of modal logic which has been deeply influenced by his work. That branch is hybrid logic, a family of logics which extend orthodox modal logic with special proposition symbols (called nominals) that name worlds. This paper explains why Henkin's techniques are so important in hybrid logic. We do so by proving a completeness result for a hybrid type theory called HTT, probably the strongest hybrid logic that has yet been explored. Our completeness result builds on earlier work with a system called BHTT, or basic hybrid type theory, and draws heavily on Henkin's work. We prove our Lindenbaum lemma using a Henkin-inspired strategy, witnessing $\Diamond$-prefixed expressions with nominals. Our use of general interpretations and the construction of the type hierarchy is (almost) pure Henkin. Finally, the generality of our completeness result is due to the first-order perspective which lies at the heart of both Henkin's best known work and hybrid logic.

1 Introduction

Leon Henkin was not a modal logician, but there is a branch of modal logic that has been deeply influenced by his work. That branch is hybrid logic, a family of logics which extend orthodox modal logic with special proposition symbols (called nominals) that name worlds. This paper explains why Henkin's influence on hybrid logic runs so deep, and we do so by proving a completeness result for a hybrid type theory that we call HTT. But before diving into higher-order logic, let's informally introduce the two central ideas of basic hybrid logic.

[*]Research partially funded by the Spanish Ministerio de Economía y Competitividad as part of the project *Logica Intensional Hibrida* (*Hybrid Intensional Logic*), FFI2013-47126-P.

The first idea is to add special propositional symbols called nominals to an orthodox modal language, and to insist that in every model, nominals are true at precisely one world. Nominals name worlds by being true there and there only. Consider this example:

$$\neg\Diamond i.$$

Here i is a nominal (nominals are conventionally written i, j, and k). Suppose we are working in a model in which the nominal i names the world w. Then this expression will be true at w if and only if w is not accessible from itself via R, the usual modal accessibility relation between worlds.

The second idea is to add modalities of the form $@_i$, where i can be any nominal. The intended semantics is written into the notation: a formula of the form $@_i\varphi$ is true at a world w if and only if φ is true at the world w' named by the nominal i. Note that a formula of the form $@_i\varphi$ is either true at all worlds, or false at all worlds: if φ is true at the world named by i, then $@_i\varphi$ is true at all worlds w; otherwise it is (everywhere) false. To put it another way, $@_i$ is a *rigidifying* operator. In any model, the world w where we evaluate $@_i\varphi$ is irrelevant: this expression always returns the truth value that φ has at the world named i.

The basic propositional hybrid language has been intensively investigated. It is decidable, indeed PSPACE-complete, just like orthodox propositional modal logic (see [2]), and it extends the expressive power of orthodox propositional modal logic (see [3] for a useful survey, and [9] for a detailed account). Most relevantly for this paper, its completeness theory is well understood; simple and general results covering many important classes of models are known (Chapter 7.3 of [4], and [7], are good starting points, and [9] is close to definitive). These results lift to first-order hybrid logic in full generality, and in [1] they were lifted to higher-order logic with the completeness proof for BHTT, basic hybrid type theory.

The BHTT system is an almost direct combination of Henkin-style type theory with basic propositional hybrid logic. But the "almost" is important, as in a higher-order setting it is natural to interpret @ as a rigidifier that can be applied not merely to formulas, but to expressions of *arbitrary* type. That is, suppose α_a is an expression of type a. In BHTT, when we evaluate $@_i\alpha_a$ at a world w, it rigidly returns the value of α_a at the world named i. This interpretation of @ is conceptually and technically appealing, and we use @ as a general rigidifier in this paper too.

HTT, the higher-order logic we work with in this paper, is BHTT enriched with the $\downarrow$ binder. Consider again the expression $\neg\Diamond i$. When working with $\downarrow$ we are free to replace nominals with state variables (typically written s and t) and to use $\downarrow$ to bind the result. So we can form:

$$\downarrow s(\neg\Diamond s).$$

So syntactically, $\downarrow$ binds out nominals.[1] But what is its semantic effect?

[1]Some authors bind nominals directly, forming expressions like $\downarrow i(\neg\Diamond i)$. There's nothing

The $\downarrow$binder works locally: it binds state variables to the world of evaluation. That is, when we evaluate an expression of the form $\downarrow s\varphi$ at a world w, the state variable s is bound to w, and all occurrences of s within the scope of $\downarrow s$ are interpreted as names for w. Consider $\downarrow s(\neg\Diamond s)$ again. When we evaluate it at w, s is bound to w, and this means that the s in $\neg\Diamond s$ is to be interpreted as a name for w. Hence $\neg\Diamond s$ is true at w if and only if w is not accessible from itself via R. So there is an important difference between $\neg\Diamond i$ and $\downarrow s(\neg\Diamond s)$. In any model, the nominal i is a *fixed* name for a world, hence $\neg\Diamond i$ only tests for irreflexivity when evaluated at the unique world named i. But $\downarrow s(\neg\Diamond s)$ binds the state variable s to the world of evaluation, hence it is an expression that tests for irreflexivity at *every* world in a model.

Basic propositional hybrid logic enriched with $\downarrow$ has also been intensively investigated. It is undecidable (see [5]), and elegant model-theoretically (see [2] and [9]). Moreover, its completeness theory is well understood, and we will build on what is known in this paper.

We proceed as follows. In Section 2 we define the syntax and semantics of HTT; we illustrate a common pattern of interaction between @ and $\downarrow$ and define substitution in detail. In Section 3 we axiomatize HTT over the class of all models. We discuss three variant axiomatizations; thinking about their differences will help us see why (and where) Henkin's techniques are important in hybrid logic. In Section 4 we prove completeness, building on the earlier proof for BHTT. As we shall see, the Lindenbaum construction for BHTT does not need to be modified; $\downarrow$ is handled automatically. In Section 5 we lift a general result from propositional hybrid logic to HTT; we show we can add certain hybrid theories as additional axioms and automatically gain completeness, and show that these axioms are equivalent in expressive power to the bounded fragment of first-order logic. In Section 6 we conclude by asking where Henkin's influence is most important in hybrid logic. We answer the question by looking more closely at a key proof rule in one of our axiomatizations.

2 Syntax and semantics of HTT

In this section we introduce the syntax and (standard and general) semantics for HTT. Our definitions are those given for BHTT in [1] extended with the clauses for $\downarrow$ and state variables (the nominal-like variables that $\downarrow$ binds). After our preliminary work, we discuss substitution and rigidity.

Syntax

Definition 1 (Syntax of HTT) ***Types****: Let t and e be two fixed objects. We define the set* TYPES *of types of* HTT *to be:*

$$\mathsf{TYPES} ::= t \mid e \mid \langle a, b\rangle \text{ with } a, b \in \mathsf{TYPES} \text{ and } a \neq t.$$

wrong with this, but it seems neater to draw a syntactic distinction between state variables (which are open to binding) and nominals (which are not).

Meaningful Expressions: *The set* ME_a *of* ***meaningful expressions of type*** *a consists of the basic and complex expressions of type a we now define.*

Basic Expressions*: For each type* $a \neq t$*, we have a denumerably infinite set* CON_a *of* ***non-logical constants*** $c_{n,a}$*, where* n *is a natural number. Constants of type* t *are* ***truth*** *and* ***falsity****, that is,* $\mathsf{CON}_t = \{\top, \bot\}$*, and we define* CON *to be* $\bigcup_a \mathsf{CON}_a$*. For each type* $a \neq t$*, we have a denumerably infinite set* VAR_a *of* ***variables*** $v_{n,a}$*, where* n *is a natural number, and we define* VAR *to be* $\bigcup_a \mathsf{VAR}_a$*. Finally, for type* t*, we have a denumerably infinite set* NOM *of nominals* i_n*, and a denumerably infinite set* SVAR *of state variables* s_n*, where* n *is a natural number. We define* HYB *to be* $\mathsf{NOM} \cup \mathsf{SVAR}$*.*

Summarizing, for each natural number n we have:

$$i_n \in \mathsf{ME}_t \mid s_n \in \mathsf{ME}_t \mid c_{n,a} \in \mathsf{ME}_a \mid v_{n,a} \in \mathsf{ME}_a \text{ with } a \neq t.$$

Complex Expressions: *These are generated as follows:*

$$(\gamma_{\langle b,a\rangle}\beta_b) \in \mathsf{ME}_a \mid (\lambda u_b \alpha_a) \in \mathsf{ME}_{\langle b,a\rangle} \mid (@_i\alpha_a) \in \mathsf{ME}_a \mid (@_s\alpha_a) \in \mathsf{ME}_a$$
$$\{(\alpha_a = \alpha'_a), (\neg\varphi_t), (\varphi_t \wedge \psi_t), (\forall u_b \varphi_t), (\Box\varphi_t), (\downarrow s\varphi_t)\} \subseteq \mathsf{ME}_t,$$

where $\alpha_a, \alpha'_a \in \mathsf{ME}_a$*,* $\beta_b \in \mathsf{ME}_b$*,* $\gamma_{\langle b,a\rangle} \in \mathsf{ME}_{\langle b,a\rangle}$*,* $u_b \in \mathsf{VAR}_b$*,* $i \in \mathsf{NOM}$*,* $s \in \mathsf{SVAR}$ *and* $\varphi_t, \psi_t \in \mathsf{ME}_t$*. As this notation illustrates, we sometimes explicitly give the type of a meaningful expression (writing, for example,* α_a *as we just did) to emphasize that* $\alpha \in \mathsf{ME}_a$*. The remaining booleans, the quantifier* $\exists$*, and the diamond* $\Diamond$*, are defined in the familiar way. We routinely drop outermost brackets, and drop others when this will not result in ambiguity. Given a set of expressions* Δ*, we define* $\mathsf{CON}(\Delta)$*,* $\mathsf{NOM}(\Delta)$*,* $\mathsf{VAR}(\Delta)$ *and* $\mathsf{SVAR}(\Delta)$ *to be (respectively) the sets of constants, nominals, variables and state variables occurring in expressions in* Δ*. We often call expressions of type t formulas.*

For those unfamiliar with propositional or first-order hybrid logic, the following point should be stressed: *nominals can occur in two distinct syntactic positions, and state variables can occur in three.* To give some simple examples, the following expressions are meaningful expressions of type t in which the nominal i and the state variable s occur in *formula position*:

$$i \qquad s \qquad i \vee \neg i \qquad \Box(s \to i) \to (\Box s \to \Box i) \qquad \Diamond\Diamond s \to \Diamond s.$$

The following are also meaningful expressions of type t, but here we see the nominal i and the state variable s also occurring in *operator position*, that is, in expressions of form $@_i$ and $@_s$ respectively:

$$@_i i \qquad @_i s \qquad @_s(i \vee \neg i) \qquad @_i(s \to i) \to (@_i s \to @_i i) \qquad @_s(\Diamond\Diamond s \to \Diamond s).$$

Finally, here are three examples in which state variables occurs in *binding position*, that is, in patterns like $\downarrow s$ and $\downarrow t$:

$$\downarrow s(\neg\Diamond s) \qquad \downarrow s(\Diamond\Diamond s \to \Diamond s) \qquad \downarrow s\Box\Box \downarrow t @_s \Diamond t.$$

In these example, all occurrences of the state variables s and t — whether in formula, operator or binder position — have been bound by the $\downarrow$ binder. Now, there are two other binders in our language, namely the familiar $\forall$ and λ binders which bind ordinary variables, so before going further let us be precise about the concepts of freedom and bondage for the three binders of HTT.

Definition 2 (Freedom and bondage) *Given a meaningful expression α, the set of **free variables** occurring in α_a (notation $\mathsf{FREE}(\alpha)$) is given by:*

$$\begin{array}{rcl}
\mathsf{FREE}(v) & = & \{v\} \textit{ for } v \in \mathsf{VAR} \\
\mathsf{FREE}(\tau) & = & \varnothing \textit{ for } \tau \in \mathsf{CON} \cup \mathsf{NOM} \cup \mathsf{SVAR} \\
\mathsf{FREE}(\alpha = \beta) = \mathsf{FREE}(\alpha\beta) = \mathsf{FREE}(\alpha \wedge \beta) & = & \mathsf{FREE}(\alpha) \cup \mathsf{FREE}(\beta) \\
\mathsf{FREE}(\neg\alpha) = \mathsf{FREE}(\Box\alpha) & = & \mathsf{FREE}(\alpha) \\
\mathsf{FREE}(@_i\alpha) = \mathsf{FREE}(@_s\alpha) = \mathsf{FREE}(\downarrow s\alpha) & = & \mathsf{FREE}(\alpha) \\
\mathsf{FREE}(\forall u\alpha) = \mathsf{FREE}(\lambda u\alpha) & = & \mathsf{FREE}(\alpha)\backslash\{u\}.
\end{array}$$

*Given a meaningful expression α, the set of **free state variables** occurring in α_a (notation $\mathsf{SFREE}(\alpha)$) is defined as follows:*

$$\begin{array}{rcl}
\mathsf{SFREE}(s) & = & \{s\} \textit{ for } s \in \mathsf{SVAR} \\
\mathsf{SFREE}(\tau) & = & \varnothing \textit{ for } \tau \in \mathsf{CON} \cup \mathsf{NOM} \cup \mathsf{VAR} \\
\mathsf{SFREE}(\alpha = \beta) = \mathsf{SFREE}(\alpha\beta) = \mathsf{SFREE}(\alpha \wedge \beta) & = & \mathsf{SFREE}(\alpha) \cup \mathsf{SFREE}(\beta) \\
\mathsf{SFREE}(\neg\alpha) = \mathsf{SFREE}(\Box\alpha) & = & \mathsf{SFREE}(\alpha) \\
\mathsf{SFREE}(\forall u\alpha) = \mathsf{SFREE}(\lambda u\alpha) & = & \mathsf{SFREE}(\alpha) \\
\mathsf{SFREE}(@_i\alpha) & = & \mathsf{SFREE}(\alpha) \\
\mathsf{SFREE}(@_s\alpha) & = & \mathsf{SFREE}(\alpha) \cup \{s\} \\
\mathsf{SFREE}(\downarrow s\alpha) & = & \mathsf{SFREE}(\alpha)\backslash\{s\}.
\end{array}$$

*If a variable v is free in a meaningful expression α, then it is bound in both $\forall v\alpha$ and $\lambda v\alpha$. Similarly, if a state variable s is free in a meaningful expression α, then it is bound in $\downarrow s\alpha$. A meaningful expression α_t of type t is called a **sentence** if and only if all the variables and state variables it contains are bound, that is, if and only if $\mathsf{FREE}(\alpha_t) \cup \mathsf{SFREE}(\alpha_t) = \varnothing$.*

Summing up: variable binding and state variable binding are distinct. Ordinary variables can only be bound by $\forall$ or λ, whereas state variables can only be bound by the $\downarrow$ binder. Moreover, it should be clear (at least syntactically) that state variables are essentially nominals open to $\downarrow$ binding, so we could have stated the syntactic clauses for nominals and state variables more compactly by stipulating that for all $h_n \in \mathsf{HYB}$, $h_n \in \mathsf{ME}_t$ and $@_h\alpha_a \in \mathsf{ME}_a$.

Nominals and $@_i$ operators are the tools characteristic of basic hybrid logic, and the BHTT system defined in [1] is built over them. The $\downarrow$ binder takes us to a richer hybrid logic, and the HTT of this paper differs from BHTT precisely by its addition. Nominals and expressions of the form $@_i\alpha_a$ (where i is any nominal) will play the central role in the completeness result for HTT: models will be built Henkin-style out of equivalence classes of witness nominals, and expressions of the form $@_i\alpha_a$ will specify how information is to be distributed. Indeed, as far as our fundamental completeness result is concerned, state variables and the $\downarrow$ binder play a rather passive role: HTT can be axiomatized by adding a single axiom schema to the axiomatization of BHTT. The $\downarrow$ binder will make its presence felt when we strengthen our fundamental completeness result.

Semantics

Definition 3 **(HTT models)** *A* ***standard structure*** *(or* ***standard model****) for HTT is a pair $\mathcal{M} = \langle \mathcal{S}, \mathsf{F} \rangle$ such that*

1. *$\mathcal{S} = \langle \langle \mathsf{D}_a \rangle_{a \in \mathsf{TYPES}}, W, R \rangle$ is a* ***standard skeleton****, where:*
   - *(a) $\langle \mathsf{D}_a \rangle_{a \in \mathsf{TYPES}}$, the* ***standard type hierarchy****, is defined as follows:*

$$\begin{array}{rcl} \mathsf{D}_t & = & \{T, F\} \text{ is the set of truth values,} \\ \mathsf{D}_e & \neq & \varnothing \text{ is the set of individuals,} \\ \mathsf{D}_{\langle a,b \rangle} & = & \mathsf{D}_b^{\mathsf{D}_a} \text{ is the set of all functions from } \mathsf{D}_a \text{ into } \mathsf{D}_b \\ & & \text{for } a, b \in \mathsf{TYPES}, a \neq t. \end{array}$$

   - *(b) $W \neq \varnothing$ is the set of worlds.*
   - *(c) $R \subseteq W \times W$ is the accessibility relation.*
   - *(d) The pair $\langle W, R \rangle$ is called a* ***frame****, and when working with a given model $\mathcal{M}$ we sometimes talk about its* ***underlying frame****.*

2. *The* ***denotation function*** *F assigns to each non-logical constant a function from W to elements of appropriate type, and assigns to each nominal a function from W to the set of truth values. More precisely:*
   - *(a) For any constant $c_{n,a}$ we define $\mathsf{F}(c_{n,a}) : W \longrightarrow \mathsf{D}_a$. Moreover, $(\mathsf{F}(\top))(w) = T$ and $(\mathsf{F}(\bot))(w) = F$, for any world $w \in W$.*
   - *(b) $\mathsf{F}(i) : W \longrightarrow \{T, F\}$ such that $(\mathsf{F}(i))(v) = T$ for a unique $v \in W$. To simplify notation, we sometimes write $\mathsf{F}(i) = \{v\}$ and say that v is the denotation of i, or the world named by i.*

Note that we are working with a constant domain semantics: we have a fixed type hierarchy $\langle \mathsf{D}_a \rangle_a \in \mathsf{TYPES}$, and F interprets all constants on this fixed domain. Furthermore, recall that a central idea of propositional hybrid logic is to use propositions as names. Because nominals are true at precisely one world in any model, they can be thought of as naming that world by being true there and there only. Our interpretation of nominals in type theory imports this basic idea to the richer setting: the interpretational constraint ensures that nominals act as world names.

We interpret ordinary variables and state variables via assignments:

Definition 4 *An* ***assignment*** *g is a function with domain $\mathsf{VAR} \cup \mathsf{SVAR}$ such that for any variable $v_{n,a}$ we have $g(v_{n,a}) \in \mathsf{D}_a$, and for any state variable s we have $g(s) \in W$.*

An assignment g' is a v-variant of g if and only if it coincides with g on all values except, perhaps, on the value assigned to the variable v. We use g_v^θ to *denote the v-variant of g whose value for $v \in \mathsf{VAR}_a$ is $\theta \in \mathsf{D}_a$. Similarly, g' is an s-variant of g if and only if it coincides with g on all values except, perhaps, on the value assigned to the state variable s. We use g^w_s to denote the s-variant of g whose value for s is $w \in W$.*

There are two things to note about this definition. First, it treats ordinary variables in the manner familiar from Henkin's work. Second, it treats state variables as syntactic entities that name worlds. An ordinary nominal is true at a unique world. An assignment maps a state variables to a unique world. State variables are essentially nominals open to binding by the $\downarrow$ binder.

Definition 5 (HTT interpretations) *A* ***standard interpretation*** *is a pair $\langle \mathcal{M}, g\rangle$, where $\mathcal{M}$ is a standard structure for* HTT *and g is a variable assignment on $\mathcal{M}$. Given a standard structure $\mathcal{M} = \langle\langle\langle \mathsf{D}_a\rangle_{a\in\mathsf{TYPES}}, W, R\rangle, \mathsf{F}\rangle$ and an assignment g we recursively define, for any meaningful expression α, the standard interpretation of α with respect to the model $\mathcal{M}$ and the assignment g, at the world w, denoted by $[[\alpha]]^{\mathcal{M},w,g}$, as follows:*

1. $[[\tau]]^{\mathcal{M},w,g} = (\mathsf{F}(\tau))(w)$, *for* $\tau \in \mathsf{CON} \cup \mathsf{NOM}$
2. $[[v_{n,a}]]^{\mathcal{M},w,g} = g(v_{n,a})$, *for* $v_{n,a} \in \mathsf{VAR}_a$
3. $[[s]]^{\mathcal{M},w,g} = T$ *if* $g(s) = w$ *and* F *if* $g(s) \neq w$, *for* $s \in \mathsf{SVAR}$
4. $[[\lambda u_b \alpha_a]]^{\mathcal{M},w,g} = f$, *where, for any* $\theta \in \mathsf{D}_b$, $f : \mathsf{D}_b \longrightarrow \mathsf{D}_a$ *is the function defined by* $f(\theta) = [[\alpha_a]]^{\mathcal{M},w,g^{\theta}_{u_b}}$
5. $[[\alpha_{\langle b,a\rangle}\beta_b]]^{\mathcal{M},w,g} = [[\alpha_{\langle b,a\rangle}]]^{\mathcal{M},w,g}([[\beta_b]]^{\mathcal{M},w,g})$
6. $[[\alpha_a = \beta_a]]^{\mathcal{M},w,g} = T$ *iff* $[[\alpha]]^{\mathcal{M},w,g} = [[\beta]]^{\mathcal{M},w,g}$
7. $[[\neg\varphi_t]]^{\mathcal{M},w,g} = T$ *iff* $[[\varphi_t]]^{\mathcal{M},w,g} = F$
8. $[[\varphi_t \wedge \psi_t]]^{\mathcal{M},w,g} = T$ *iff* $[[\varphi_t]]^{\mathcal{M},w,g} = T$ *and* $[[\psi_t]]^{\mathcal{M},w,g} = T$
9. $[[\forall x_a \varphi_t]]^{\mathcal{M},w,g} = T$ *iff for all* $\theta \in \mathsf{D}_a$ $[[\varphi]]^{\mathcal{M},w,g^{\theta}_{x_a}} = T$
10. $[[\Box\varphi_t]]^{\mathcal{M},w,g} = T$ *iff for all* $v \in W$ *such that* $\langle w, v\rangle \in R$, $[[\varphi_t]]^{\mathcal{M},v,g} = T$
11. $[[@_i\alpha_a]]^{\mathcal{M},w,g} = [[\alpha_a]]^{\mathcal{M},v,g}$ *where* $\{v\} = \mathsf{F}(i)$.
12. $[[@_s\alpha_a]]^{\mathcal{M},w,g} = [[\alpha_a]]^{\mathcal{M},v,g}$ *where* $v = g(s)$.
13. $[[\downarrow s\varphi_t]]^{\mathcal{M},w,g} = T$ *iff* $[[\varphi_t]]^{\mathcal{M},w,g^w_s} = T$

Some remarks on the clauses covering nominal and state variables. Consider Clause 1 when τ is a nominal. This covers occurrences of i in formula position, and in such cases i should be true at precisely the world it denotes. Because of the constraint on the interpretation of nominals, this is what Clause 1 gives us. Next consider Clause 11, which covers occurrences of i in operator position. We want $@_i\alpha$ to be an expression (of the same type as α) that rigidly yields the value of α at the world named by i. Clause 11 gives us this.

Now for state variables. Clause 3 covers occurrences of s in formula position. State variables are assigned a unique world, the world they name. As they are expression of type t, they should be true at that world and at that world only, which is what Clause 3 insists upon. Clause 12 deals with s in operator position, and says that $@_s\alpha_a$ is true at a world w if and only if α_a is true at v, the world named by s. This mirrors the clause for nominals in operator position, and indeed we could collapse Clauses 11 and 12 together by stating that for all $h \in \mathsf{HYB}$, $[[@_h\alpha_a]]^{\mathcal{M},w,g}$ is $[[\alpha_a]]^{\mathcal{M},v,g}$, where v is the world named by h.

But state variables were only introduced to support the $\downarrow$ binder that distinguishes HTT from BHTT, so Clause 13 is the real novelty. This says that $\downarrow s\varphi$ binds s to the world of evaluation, and that all occurrences of s in φ within its scope are to be interpreted as names for this world. So to speak, $\downarrow s$ creates a temporary name s for the world of evaluation. Consider again the expression $\downarrow s(\neg\Diamond s)$ we discussed in the introduction. As we said there, $\downarrow$ binds s to the world of evaluation. The semantics just defined guarantees that if we evaluate this expression at any world w in any model, it will be true precisely when w is not accessible (via R) from itself.

Let's look at a second example, which illustrates a common theme: $\downarrow$ binding a state variable occurring in an @ operator under its scope. Let Woman be an expression of type $\langle e, t\rangle$ which picks out the women in each possible world, and let Potus be an expression of type e which picks out the President of the United States in each possible world. Suppose that an American voter murmurs: *The President of the United Sates might be a woman.* We'll consider three readings of this sentence. The first is that the voter is thinking about a possible world in which Barack Obama is a woman:

$$\downarrow s\Diamond(\mathsf{Woman}(@_s\mathsf{Potus})).$$

This expression is correctly typed: Potus is of type e, hence so is $@_s\mathsf{Potus}$, hence Woman can be applied to it yielding the type t expression $\mathsf{Woman}(@_s\mathsf{Potus})$. Prefixing this with $\downarrow s\Diamond$ yields the above sentence. This expresses the first reading of the utterance: s is bound to the utterance world, which means that the embedded expression $@_s\mathsf{Potus}$ must be evaluated at the utterance world too, yielding the value Obama. So the voter is musing about what it might be like in a possible world in which Obama is a woman.

On the other hand, maybe our voter simply meant this:

$$\Diamond(\mathsf{Woman\ Potus}).$$

That is, perhaps our voter merely meant that there was some possible world in which the president in that world (whoever she may be) is a woman. Well, perhaps. Indeed if our voter had lived in (say) 1950, this reading may have been all that the voter considered possible.

But it is also possible (indeed, nowadays more likely) that our voter is musing about the growing number of powerful American woman politicians, and means that: *The President of the United Sates might be a women existing in the utterance world.* We can express this by:

$$\downarrow s\Diamond((@_s\mathsf{Woman})\ \mathsf{Potus}).$$

This is also correctly typed: Woman is of type $\langle e, t\rangle$, hence so is $@_s\mathsf{Woman}$, and so this can be applied to the type e expression Potus. This yields an expression of type t, and, as before, prefixing $\downarrow s\Diamond$ gives us a sentence. But note the difference: the bound state variable s forces Woman to be evaluated at the utterance world, hence the sentence is true if and only if there is an individual in some possible world who is president there, and a woman in the utterance world.

This example illustrates an important point: @ and $\downarrow$ are a powerful team. The @ operator is a rigidifying operators for all types: it universalizes a local value. The $\downarrow$ binder enables us to force evaluation of state variable at the current world: it is a *localizing* binder. And when, as in the Potus example, an occurrence of $\downarrow s$ is used to 'store' a value for s which is later 'retrieved' by occurrences of $@_s$ under its scope, we are able to shift the evaluation of embedded expressions to the world named s, and hence to draw interesting distinctions.

General semantics

The standard semantics for higher-order logic is strong: if we define validity as truth in all standard structures, then the set of validities cannot be axiomatized. In 1950 Henkin proposed a weaker notion of interpretation for higher-order logic (see Henkin [13, 14]). As he showed, defining validity with respect to *general interpretations* lowers the logical complexity of validity (as there are more generalized structures than standard ones, it becomes easier to falsify a formula, so there are fewer validities) and this new notion of validity can be axiomatized in a natural way. We follow Henkin's approach and prove our completeness results with respect to general interpretations.

Definition 6 (HTT skeletons and structures) *A* ***type hierarchy*** *is a family $\langle \mathsf{D}_a\rangle_{a\in\mathsf{TYPES}}$ of sets defined recursively as follows:*

$$\begin{array}{rcl} \mathsf{D}_e & \neq & \varnothing \\ \mathsf{D}_t & = & \{T, F\} \\ \mathsf{D}_{\langle a,b\rangle} & \subseteq & \mathsf{D}_b^{\mathsf{D}_a} \textit{ for } a, b \in \mathsf{TYPES}, a \neq t. \end{array}$$

A ***skeleton*** $\mathcal{S} = \langle\langle \mathsf{D}_a\rangle_{a\in\mathsf{TYPES}}, W, R\rangle$ *is a triple satisfying all the conditions of a standard skeleton except that $\langle \mathsf{D}_a\rangle_{a\in\mathsf{TYPES}}$ is a (not necessarily standard) type hierarchy. A* ***structure*** *(or* ***model****) is a pair $\mathcal{M} = \langle \mathcal{S}, \mathsf{F}\rangle$ where $\mathcal{S}$ is a skeleton and F is a denotation function.*

This definition encapsulates the idea that we don't need all the functions from D_a to D_b to interpret expressions of type $\langle a, b\rangle$. However we do need enough functions to interpret all the expressions of our language, which motivates the following:

Definition 7 (General interpretation) *A* ***general interpretation*** *is a pair $\langle \mathcal{M}, g\rangle$ where $\mathcal{M}$ is a structure, g a variable assignment, and for any meaningful expression in ME_a, its interpretation (as given by Definition 5) is in D_a.*

That is, from now on, given a (not necessarily standard) model $\mathcal{M}$, an assignment g, and an expression α, we will interpret α on $\mathcal{M}$ using the clauses given in Definition 5.

We can now define consequence and validity. Note that these definitions really do generalize the standard ones, for every standard interpretation is a generalized interpretation (but not conversely).

Definition 8 (Consequence and validity) *Let $\Gamma \cup \{\varphi\} \subseteq \mathsf{ME}_t$ and $\mathcal{M}$ be a structure. We define consequence and validity as follows:*

Consequence: *We say that φ is a consequence of Γ, written $\Gamma \models \varphi$, if and only if for all general interpretations $\langle \mathcal{M}, g \rangle$ and all $w \in W$, if $[[\gamma]]^{\mathcal{M},w,g} = T$ for all $\gamma \in \Gamma$ then $[[\varphi]]^{\mathcal{M},w,g} = T$.*

Validity: *We say that φ is valid, written $\models \varphi$, if and only if φ is a consequence of the empty set (that is $\varnothing \models \varphi$).*

A useful way of thinking about generalized interpretation is as a mechanism that reduces higher-order logic to first-order logic, or (perhaps better) as a mechanism that picks out the higher-order validities that are essentially first-order from the rich space of standard higher-order validities; for useful discussions, see [16] and [15]. We mention this because we are going to use Henkin's method of constants to prove a completeness result, Theorem 33, that covers a wide range of frame classes. From an orthodox modal perspective, this result is atypically general. But when viewed from the first-order perspective that underpins Henkin's work, its generality is natural, for as we shall see in Section 5, the hybrid machinery used in this paper is essentially first-order. Indeed the pure nominal-free sentential fragment we shall discuss there is essentially hybrid notation for the bounded fragment of the first-order language of frames.

Substitution and rigidity

To conclude this section, some syntactic lemmas concerning substitution and rigidity. Our first lemma is Lemma 9 from [1] extended to cover state variables. It is proved by induction on the structure of terms, and we leave it to the reader. The inductive steps for λ and $\forall$ can be found in [1].

Lemma 9 (Agreement for variables and state variables) *Let g and h be assignments that agree on the values assigned to the free variables and state variables of α; that is, f and g agree on the values they assign to all the elements of $\mathsf{FREE}(\alpha_a) \cup \mathsf{SFREE}(\alpha_a)$. Let $\langle \mathcal{M}, g \rangle$ and $\langle \mathcal{M}, h \rangle$ be general interpretations. Then for any world w we have that $[[\alpha_a]]^{\mathcal{M},w,g} = [[\alpha_a]]^{\mathcal{M},w,h}$.*

We now define substitution. We first deal with substitution for state variables. This is simpler than ordinary variable substitution, as the only expressions substitutable for a free state variable are nominals and state variables, that is, elements h of HYB. Any such h is a basic expression, and as neither nominals nor state variables can be bound by λ or $\forall$, we do not have to worry about accidental binding from these sources. So we need merely specify how any $h \in \mathsf{HYB}$ should be substituted into formula, operator, and binding positions.

Definition 10 (State variable substitution) *Let $h \in \mathsf{HYB}$. We define, for all $\alpha_a \in \mathsf{ME}_a$, the **substitution of** h **for a state variable** s **in** α_a, written $\alpha_a(\frac{h}{s})$, as follows:*

1. $\tau(\frac{h}{s}) := \tau$ *for* $\tau \in \mathsf{CON} \cup \mathsf{VAR} \cup \mathsf{NOM}$
2. $s'(\frac{h}{s}) := \begin{cases} h & \textit{if } s' = s \\ s' & \textit{if } s' \neq s \end{cases}$
3. $(\lambda u_p \beta_b)(\frac{h}{s}) := \lambda u_p(\beta_b(\frac{h}{s}))$
4. $(\beta_{\langle b,a \rangle}\delta_b)(\frac{h}{s}) := \beta_{\langle b,a \rangle}(\frac{h}{s})\delta_b(\frac{h}{s}) \mid (\beta_b = \delta_b)(\frac{h}{s}) := \beta_b(\frac{h}{s}) = \delta_b(\frac{h}{s})$
5. $(\neg\varphi)(\frac{h}{s}) := \neg(\varphi(\frac{h}{s})) \mid (\varphi \wedge \psi)(\frac{h}{s}) := \varphi(\frac{h}{s}) \wedge \psi(\frac{h}{s})$
6. $(\forall u_p \beta_b)(\frac{h}{s}) := \forall u_p(\beta_b(\frac{h}{s}))$
7. $(\Box\psi)(\frac{h}{s}) := \Box(\psi(\frac{h}{s}))$
8. $(@_i \beta_b)(\frac{h}{s}) := @_i(\beta_b(\frac{h}{s}))$
9. $(@_{s'} \beta_b)(\frac{h}{s}) := \begin{cases} @_h(\beta_b(\frac{h}{s})) & \textit{if } s' = s \\ @_{s'}(\beta_b(\frac{h}{s})) & \textit{if } s' \neq s \end{cases}$
10. $(\downarrow s' \beta_b)(\frac{h}{s}) := \begin{cases} \downarrow s' \beta_b & \textit{if } s \notin \mathsf{SFREE}(\downarrow s' \beta_b) \\ \downarrow s' \beta_b & \textit{if } s \in \mathsf{SFREE}(\downarrow s' \beta_b) \textit{ and } h = s' \\ \downarrow s'(\beta_b(\frac{h}{s})) & \textit{if } s \in \mathsf{SFREE}(\downarrow s' \beta_b) \textit{ and } h \neq s' \end{cases}$

With state variable substitution $(\frac{h}{s})$ at our disposal, we can define ordinary variable substitution $(\frac{\gamma_c}{v_c})$, the substitution of a (possibly complex) expression γ_c for a free variable of type c. As γ_c may be complex, accidental binding is an issue. However it is a well understood issue. The following definition is (for the most part) standard: it uses the usual type-theoretic definitions that prevent accidental binding of ordinary variables by λ and $\forall$. Only the $\downarrow$ clause requires comment. As we know, we can freely substitute ordinary variables under the scope of the $\downarrow$ binder; accidental binding of ordinary variables in γ_c by $\downarrow$ is impossible. But γ_c may contain free occurrences of the state variable s, and we must prevent $\downarrow s$ from accidentally binding these. But this is easily done: we need merely make use of state variable substitution $(\frac{h}{s})$ as just defined.

Definition 11 (Variable substitution) *For all $\alpha_a \in \mathsf{ME}_a$, the **substitution of** γ_c **for a variable** v_c **in** α_a, written $\alpha_a(\frac{\gamma_c}{v_c})$, is defined as follows:*

1. $\tau(\frac{\gamma_c}{v_c}) := \tau$ *for* $\tau \in \mathsf{CON} \cup \mathsf{NOM} \cup \mathsf{SVAR}$

2. $v_a(\frac{\gamma_c}{v_c}) := \begin{cases} \gamma_c & \textit{if } v_a = v_c \\ v_a & \textit{if } v_a \neq v_c \end{cases}$

3. $(\lambda u_p\beta_b)(\frac{\gamma_c}{v_c}) := \begin{cases} \lambda u_p\beta_b & \textit{if } v_c \notin \mathsf{FREE}(\lambda u_p\beta_b) \\ \lambda u_p(\beta_b(\frac{\gamma_c}{v_c})) & \textit{if } v_c \in \mathsf{FREE}(\lambda u_p\beta_b), \\ & \quad u_p \notin \mathsf{FREE}(\gamma_c) \\ (\lambda x_p(\beta_b\frac{x_p}{u_p}))(\frac{\gamma_c}{v_c}) & \textit{if } v_c \in \mathsf{FREE}(\lambda u_p\beta_b), \\ & \quad u_p \in \mathsf{FREE}(\gamma_c), x_p \textit{ new} \end{cases}$

4. $(\beta_{\langle b,a\rangle}\delta_b)(\frac{\gamma_c}{v_c}) := \beta_{\langle b,a\rangle}(\frac{\gamma_c}{v_c})\delta_b(\frac{\gamma_c}{v_c}) \mid (\beta_b = \delta_b)(\frac{\gamma_c}{v_c}) := \beta_b(\frac{\gamma_c}{v_c}) = \delta_b(\frac{\gamma_c}{v_c})$

5. $(\neg\varphi)(\frac{\gamma_c}{v_c}) := \neg(\varphi(\frac{\gamma_c}{v_c})) \mid (\varphi \wedge \psi)(\frac{\gamma_c}{v_c}) := \varphi(\frac{\gamma_c}{v_c}) \wedge \psi(\frac{\gamma_c}{v_c})$

6. $(\forall u_p\psi)(\frac{\gamma_c}{v_c}) := \begin{cases} \forall u_p\psi & \textit{if } v_c \notin \mathsf{FREE}(\forall u_p\psi) \\ \forall u_p(\psi(\frac{\gamma_c}{v_c})) & \textit{if } v_c \in \mathsf{FREE}(\forall u_p\psi), \\ & \quad u_p \notin \mathsf{FREE}(\gamma_c) \\ (\forall x_p(\psi\frac{x_p}{u_p}))(\frac{\gamma_c}{v_c}) & \textit{if } v_c \in \mathsf{FREE}(\forall u_p\psi), \\ & \quad u_p \in \mathsf{FREE}(\gamma_c), x_p \textit{ new} \end{cases}$

7. $(\Box\psi)(\frac{\gamma_c}{v_c}) := \Box(\psi(\frac{\gamma_c}{v_c}))$

8. $(@_i\beta_b)(\frac{\gamma_c}{v_c}) := @_i(\beta_b(\frac{\gamma_c}{v_c}))$.

9. $(@_s\beta_b)(\frac{\gamma_c}{v_c}) := @_s(\beta_b(\frac{\gamma_c}{v_c}))$.

10. $(\downarrow s\psi)(\frac{\gamma_c}{v_c}) := \begin{cases} \downarrow s\psi & \textit{if } v_c \notin \mathsf{FREE}(\downarrow s\psi) \\ \downarrow s\psi(\frac{\gamma_c}{v_c}) & \textit{if } v_c \in \mathsf{FREE}(\downarrow s\psi), \\ & \quad s \notin \mathsf{SFREE}(\gamma_c) \\ (\downarrow t(\psi\frac{t}{s}))(\frac{\gamma_c}{v_c}) & \textit{if } v_c \in \mathsf{FREE}(\downarrow s\psi), \\ & \quad s \in \mathsf{SFREE}(\gamma_c), t \textit{ new} \end{cases}$

We now define ***rigid expressions***. These are expressions that have the same value at all worlds. They play an important role in our axiomatization, and equivalence classes of rigid expressions are the Lego bricks of the Henkin-style type hierarchy construction we use in the completeness proof.

The following definition is Definition 11 of [1] extended to cover state variables in both formula and operator position.

Definition 12 (Rigid expressions) *The set* RIGIDS*, consisting of **rigid meaningful expressions**, is defined as follows:*

$$\mathsf{RIGIDS} ::= \bot \mid \top \mid h \mid v_a \mid @_h\theta_a \mid \lambda v_b\alpha_a \mid \gamma_{\langle b,a\rangle}\beta_b \mid \alpha_b = \beta_b \mid \neg\varphi_t \mid \varphi_t \wedge \psi_t \mid \forall v_a\varphi_t,$$

where $h \in \mathsf{HYB}$*,* $\theta_a \in \mathsf{ME}_a$ *and* α_a*,* β_b*,* $\gamma_{\langle b,a\rangle}$*,* φ_t*,* $\psi_t \in \mathsf{RIGIDS}$*. We say that* $\alpha \in \mathsf{RIGIDS}_a$ *if* α *is rigid and of type* a*, that is, if* $\alpha \in \mathsf{RIGIDS} \cap \mathsf{ME}_a$*.*

Unsurprisingly, $\downarrow$ is conspicuous by its absence, for $\downarrow$ does not rigidify. As a simple example, consider the sentence $\downarrow s(\Diamond s)$. In any model containing both reflexive and irreflexive worlds, this sentence is true at all the reflexive worlds and false at the rest.

Lemma 13 *Let $\langle \mathcal{M}, g\rangle$ be a general interpretation. Then for any $\gamma \in \mathsf{RIGIDS}$, we have that $[[\gamma]]^{\mathcal{M},w,g} = [[\gamma]]^{\mathcal{M},v,g}$ for all $w, v \in W$.*

Proof By induction on the construction of rigid expressions. The steps for $\lambda v_b \alpha_a$ and $\forall v_a \varphi$ can be found the proof of Lemma 12 in [1]. $\dashv$

Rigid expressions behave straightforwardly with respect to substitution:

Lemma 14 (Rigid substitution) *Let $\langle \mathcal{M}, g\rangle$ be a general interpretation. Then for all worlds w, all $\alpha_a \in \mathsf{ME}_a$, all $h \in \mathsf{HYB}$ and all state variables s:*

$$[[\alpha_a(\frac{h}{s})]]^{\mathcal{M},w,g} = [[\alpha_a]]^{\mathcal{M},w,g^{\overline{h}}_{s}}$$

where $\overline{h}$ is an abbreviation for $[[h]]^{\mathcal{M},w,g}$, that is, it is the world named by h.

Furthermore, for all worlds w, all $\alpha_a \in \mathsf{ME}_a$, all meaningful expressions γ_c of type c, and all variables $v_c \in \mathsf{VAR}_c$:

$$[[\alpha_a(\frac{\gamma_c}{v_c})]]^{\mathcal{M},w,g} = [[\alpha_a]]^{\mathcal{M},w,g^{\overline{\gamma_c}}_{v_c}}$$

where $\overline{\gamma_c}$ is an abbreviation for $[[\gamma_c]]^{\mathcal{M},w,g}$.

Proof By induction on the construction of expressions. Use Lemma 9. $\dashv$

3 Axiomatizing HTT

We shall now axiomatize HTT by adding a single axiom schema to the BHTT axiomatization of [1]. We call this axiomatization K1. At the end of this section we note two more axiomatizations, K2 and K3.[2] The differences between these systems are unimportant as far as our technical results are concerned, but at the end of the paper we will discuss what K2 tells us about Henkin's influence in hybrid logic.

Axioms

As axioms we take all HTT instances of propositional tautologies together with all HTT instances of the following schemas; we use h and h' as metavariables over elements of HYB:

1. **Distributivity schemas:**
   (a) **$\Box$-distributivity:** $\vdash \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$.
   (b) **@-distributivity:** $\vdash @_h(\varphi \to \psi) \to (@_h\varphi \to @_h\psi)$.
   (c) **$\forall$-distributivity:** $\vdash \forall x_b(\varphi \to \psi) \to (\forall x_b\varphi \to \forall x_b\psi)$.

[2]In modal logic, the basic proof system is usually called K in honor of Saul Kripke. The three axiomatizations considered here are alternative ways of providing a basic proof system for HTT.

2. **Quantifier schemas:**
   (a) **$\forall$-elimination:** For β_b rigid, $\vdash \forall x_b\varphi \to \varphi(\frac{\beta_b}{x_b})$.
   (b) **Vacuous:** $\vdash \varphi \to \forall y_a\varphi$, where y_a does not occur free in φ.
3. **Equality schemas:**
   (a) **Reflexivity:** $\vdash \alpha_a = \alpha_a$.
   (b) **Substitution:** For α_a, β_a rigid, $\vdash \alpha_a = \beta_a \to (\delta_c(\frac{\alpha_a}{v_a}) = \delta_c(\frac{\beta_a}{v_a}))$.
4. **Functional schemas:**
   (a) **Extensionality:** $\vdash \forall v_b(\gamma_{\langle b,a\rangle}v_b = \delta_{\langle b,a\rangle}v_b) \to \gamma_{\langle b,a\rangle} = \delta_{\langle b,a\rangle}$, where v_b does not occur free in $\gamma_{\langle b,a\rangle}$ or $\delta_{\langle b,a\rangle}$.
   (b) **β-conversion:** For rigid β_b, $\vdash (\lambda x_b\alpha_a)\beta_b = \alpha_a(\frac{\beta_b}{x_b})$.
   (c) **η-conversion:** $\vdash (\lambda x_b\gamma_{\langle b,a\rangle}x_b) = \gamma_{\langle b,a\rangle}$, where x_b is not free in $\gamma_{\langle b,a\rangle}$.
5. **Basic hybrid schemas:**
   (a) **Selfdual:** $\vdash @_h\varphi \leftrightarrow \neg @_h\neg\varphi$.
   (b) **Intro:** $\vdash h \to (\varphi \leftrightarrow @_h\varphi)$.
   (c) **Back:** $\vdash \Diamond @_h\varphi \to @_h\varphi$.
   (d) **Ref:** $\vdash @_h h$.
   (e) **Agree:** $\vdash @_{h'}@_h\alpha_a = @_h\alpha_a$.
6. **Domain schema:**
   (a) **Hybrid Barcan:** $\vdash \forall x_b @_h\varphi \leftrightarrow @_h\forall x_b\varphi$.
7. **Basic hybrid type theory schemas:**
   (a) **Equality-at-named-worlds:** $\vdash @_h(\beta_b = \delta_b) = (@_h\beta_b = @_h\delta_b)$.
   (b) **Rigid function application:** $\vdash @_h(\gamma_{\langle b,a\rangle}\beta_b) = (@_h\gamma_{\langle b,a\rangle})(@_h\beta_b)$.
   (c) **Rigids are rigid:** If α_a is rigid then $\vdash @_h\alpha_a = \alpha_a$.
8. **Downarrow schema:**
   (a) **DA:** $\vdash @_i(\downarrow s\varphi \leftrightarrow \varphi(\frac{i}{s}))$.

The first seven groups of schemas are those used in [1] to axiomatize BHTT. In the present paper, these schemas range over HTT expressions (not merely BHTT expressions) and operators of the form $@_i$ and $@_j$ have been replaced by operators of the form $@_h$ and $@_{h'}$ to reflect the addition of state variables, but otherwise the systems are identical. Moreover, with the exception of the **Basic hybrid type theory schemas**, which were introduced in [1], these schemas are familiar from either higher-order logic or hybrid logic.

Now, to move from BHTT to HTT, we need only add the **DA** schema. Readers familiar with hybrid logic will recognize this as a standard schema used to prove completeness when $\downarrow$ is added to basic (propositional or first-order) hybrid logic. The **DA** schema spells out the locality of $\downarrow$ with admirable precision: when we are working at a world named i, we are free (reading the $\leftrightarrow$ in the left to right direction) to eliminate $\downarrow$ by substituting i for bound occurrences of s, or (reading it in the right to left direction) to use $\downarrow s$ to bind out occurrences of i. Because this schema captures the local semantics of $\downarrow$ so cleanly, the completeness proof for BHTT needs only relatively minor modifications to extend it to HTT.

Rules of proof

1. **Modus Ponens:** If $\vdash \varphi$ and $\vdash \varphi \to \psi$, then $\vdash \psi$.
2. **Generalizations:**
   (a) **Gen$_\Box$:** If $\vdash \varphi$, then $\vdash \Box\varphi$.
   (b) **Gen$_@$:** If $\vdash \varphi$, then $\vdash @_h\varphi$.
   (c) **Gen$_\forall$:** If $\vdash \varphi$, then $\vdash \forall x_a\varphi$.
3. **Rigid replacement:** If $\vdash \varphi$, then $\vdash \varphi'$, where φ' is obtained from φ by:
   (a) Uniformly replacing expressions $h \in \mathsf{HYB}$ by expressions $h' \in \mathsf{HYB}$.
   (b) Uniformly replacing variables of type a by rigid expressions of type a.
4. **Name:** If $\vdash @_h\varphi$ and h does not occur in φ, then $\vdash \varphi$.
5. **Bounded Generalization:** If $\vdash @_h\Diamond j \to @_j\varphi$ and $j \neq h$ and j does not occur in φ, then $\vdash @_h\Box\varphi$.

These are standard rules from hybrid and classical logic. The restriction in the rigid replacement rule (that only nominals and state variables can replace nominals and state variables) reflects the fact that nominals and state variables are names for worlds, and replacement must respect this. The additional restriction (that variables can only be freely replaced by rigid terms) reflects the fact that assignment functions interpret variables rigidly, and replacement must respect this too. We'll discuss **Name** and **Bounded Generalization** shortly when we consider alternative axiomatizations.

Definition 15 *A **proof** of φ is a finite sequence $\alpha_1, \ldots, \alpha_n$ of expressions such that $\alpha_n := \varphi$ and for every $1 \leq i \leq n-1$, either α_i is an axiom, or α_i is obtained from previous expressions in the sequence using the rules of proof. We write $\vdash \varphi$ whenever we have such a sequence and say that φ is a HTT-**theorem**.*

Definition 16 *If $\Gamma \cup \{\varphi\}$ is a set of meaningful expressions of type t, a **proof of** φ **from** Γ is a proof of $\vdash \gamma_1 \wedge \ldots \wedge \gamma_n \to \varphi$ where $\{\gamma_1, \ldots, \gamma_n\} \subseteq \Gamma$. A meaningful expression φ of type t is **provable from a set of expressions** Γ, written $\Gamma \vdash \varphi$, if and only if there is a proof of φ from Γ. We call Γ **inconsistent** if and only if for all formulas φ, $\Gamma \vdash \varphi$. Otherwise Γ is **consistent**.*

Theorem 17 (Soundness) *For all $\varphi \in \mathsf{ME}_t$, we have $\vdash \varphi$ implies $\models \varphi$.*

Proof Straightforward but tedious. $\dashv$

That was $\mathsf{K1}$, simply the BHTT axiomatization enriched with the **DA** schema. But more should be said about the **Name** and **Bounded generalization** rules. Neither is an orthodox modal rule of proof. However both draw on familiar ideas from classical logic, indeed both rules can be viewed as (axiomatic simulations of) natural deduction rules. Consider **Name**. This can be read as saying:

> *If φ can be proved to hold at an arbitrary world h not mentioned in φ, then we can (so to speak) discharge the $@_h$ and prove φ.*

And **Bounded generalization** can be read similarly:

> *If φ can be proved to hold at some arbitrary successor world j of h, which is not mentioned in φ, then we can (so to speak) discharge the $@_h\Diamond j$ assumption and prove that φ occurs at all successors of h.*

Natural deduction systems for hybrid logic have been intensively studied (for a monograph-length treatment, see [8]) and the ideas just sketched are central to many such proof systems.

And this leads us to K2. This is the axiomatization obtained by discarding **Bounded generalization** and replacing it with the **Paste** rule:

$$\frac{\vdash @_i\Diamond j \wedge @_j\varphi \to \theta}{\vdash @_i\Diamond\varphi \to \theta}\ .$$

That is, we replace one non-orthodox rule by another (note: we retain **Name**). The **Paste** rule is interesting for two reasons. Firstly, as we shall see when we discuss the Lindenbaum construction, this is the rule that most directly licenses the use of Henkin-style witness nominals. Secondly, as we shall discuss at the end of the paper, **Paste** is essentially a lightly disguised tableau rule. But let's defer further discussion of the second option and turn to the third.

We obtain K3 when we discard both **Name** and **Paste** and replace them with the following axioms and rules:

Gen$_\downarrow$: if $\vdash\varphi$ then $\vdash\downarrow s\varphi$.
Name$_\downarrow$: $\vdash\downarrow s(s\to\varphi)\to\varphi$, where s does not occur in φ
Bounded Generalization$_\downarrow$: $\vdash @_i\Box\downarrow s @_i\Diamond s$

This approach (due to Balder ten Cate) is elegant. The new rule is orthodox; indeed, it mirrors our other **Gen** rules. And the two new axioms show that when we have $\downarrow$ in our language, the proof-theoretic effect of the **Name** and **Bounded generalization** rules can be captured by axioms.[3]

4 Completeness

We now prove the completeness of our axiomatization(s). The involves only minor modifications of the completeness proof for BHTT. We sketch what is required, highlighting issues involving the $\downarrow$ binder.

[3]These three approaches have a common evolutionary history. The earliest was the **Paste** plus **Name** combination found in K2. This was introduced in [7] in the setting of propositional tense logic with $\downarrow$ and used to prove analogs of this paper's Theorems 29 and 33. The **Bounded generalization** plus **Name** combination used in K1 was designed to provide a natural deduction style counterpart to the tableau style **Paste** rule. Balder ten Cate then showed how this combination could be refined (when $\downarrow$ is in the language) with the rules and axioms used in K3. For detailed explorations of the **Bounded generalization** plus **Name** option, and their simplifications with $\downarrow$ see [6] and [9]. Another non-orthodox hybrid proof rule (one which is useful even if we don't have @ in our language) is the **COV** rule of [11].

The Lindenbaum construction

As with any Henkin proof, we shall build our model out of the expressions contained in a maximal consistent set of formulas, that is, out of the items in some $\Delta \subseteq \mathsf{ME}_t$. Moreover, we are going to build the maximal consistent sets we require using a Henkin-style strategy. That means that each $\exists$-formula will be witnessed by a constant of appropriate type, and we will build the functional hierarchy out of equivalence classes of these elements; this part of our proof follows Henkin's recipe almost to the letter. But working Henkin-style in hybrid logic also means that we are going to use our hybrid machinery to imitate Henkin's strategy in the modal part of the language: each $\Diamond$-formula will be witnessed by a nominal, and the worlds will be built out of equivalence classes of witness nominals. This motivates the following definition:

Definition 18 *Let Σ be a set of meaningful expressions.*

1. *Σ is* named *iff one of its elements is a nominal.*
2. *Σ is* $\Diamond$-saturated *iff for all expressions $@_i\Diamond\varphi \in \Sigma$ there is a witness nominal; that is, a nominal $j \in \mathsf{NOM}$ such that $@_i\Diamond j \in \Sigma$ and $@_j\varphi \in \Sigma$.*
3. *Σ is* $\exists$-saturated *iff for all expressions $@_i\exists x_a\varphi \in \Sigma$ there is a witness constant; that is, a constant $c_a \in \mathsf{CON}_a$ such that $@_i\varphi(\frac{@_i c_a}{x_a}) \in \Sigma$.*

So far so good — but what about the $\downarrow$ binder? If $\exists$ requires witness constants, does not the $\downarrow$ binder, like $\Diamond$, require witness nominals? Don't we also need our maximal consistent sets to be $\downarrow$-saturated? The answer is: yes, we do, but this is given to us automatically, courtesy of the **DA** schema.

Lemma 19 *Let Δ be maximal consistent, and let i be any nominal such that $i \in \Delta$. Then we have that $@_i{\downarrow}s\varphi \in \Delta$ iff $@_i\varphi(\frac{i}{s}) \in \Delta$.*

Proof Recall that **DA** is $\vdash @_i({\downarrow}s\varphi \leftrightarrow \varphi(\frac{i}{s}))$. Hence if $@_i{\downarrow}s\varphi \in \Delta$, then by the left-to-right direction we have that $\vdash @_i\varphi(\frac{i}{s}) \in \Delta$. Conversely, if $\vdash @_i\varphi(\frac{i}{s}) \in \Delta$, then by the right-to-left direction we have that $@_i{\downarrow}s\varphi \in \Delta$. $\dashv$

The $\downarrow$ operator, though expressively powerful, is deductively straightforward because of its locality. Unlike $\exists$ and $\Diamond$ which need *new* witness constants and nominals, we can specify in advance the witnesses that $\downarrow$ needs, and **DA** does this. It tells us that, at a world named i, we are free to use i as a $\downarrow$ witness, and furthermore, that we can also use $\downarrow$ to bind out i. Because of this, the Lindenbaum construction for HTT is identical to the construction for BHTT, as we don't need to modify the construction to cope with the $\downarrow$ binder.

Lemma 20 (Lindenbaum) *Let Σ be a consistent set of formulas. Then Σ can be extended to a maximal consistent set Σ^ω that is named, $\Diamond$-saturated and $\exists$-saturated.*

Proof Let $\{i_n\}_{n\in\omega}$ be an enumeration of a countably infinite set of new nominals, $\{c_{n,a}\}_{n\in\omega}$ an enumeration of a countably infinite set of new constants of type a, and $\{\varphi_n\}_{n\in\omega}$ an enumeration of the formulas of the extended language. We will build $\{\Sigma^n\}_{n\in\omega}$, a family of subsets of ME_t, by induction:

- $\Sigma^0 = \Sigma \cup \{i_0\}$.
- Assume that Σ^n has already been built. To define Σ^{n+1} we distinguish four cases:
  1. $\Sigma^{n+1} = \Sigma^n$, if $\Sigma^n \cup \{\varphi_n\}$ is inconsistent.
  2. $\Sigma^{n+1} = \Sigma^n \cup \{\varphi_n\}$, if $\Sigma^n \cup \{\varphi_n\}$ is consistent and φ_n is not of the form $@_i\Diamond\psi$ or $@_i\exists x_a\psi$.
  3. $\Sigma^{n+1} = \Sigma^n \cup \{\varphi_n, @_i\Diamond i_m, @_{i_m}\psi\}$, if $\Sigma^n \cup \{\varphi_n\}$ is consistent, φ_n has the form $@_i\Diamond\psi$, and i_m is the first nominal not in Σ^n or φ_n.
  4. $\Sigma^{n+1} = \Sigma^n \cup \{\varphi_n, @_i\psi\frac{@_i c_{m,a}}{x_a}\}$, if $\Sigma^n \cup \{\varphi_n\}$ is consistent, φ_n has the form $@_i\exists x_a\psi$, and $c_{m,a}$ is the first constant of type a not in Σ^n or φ_n.

Now, let $\Sigma^\omega = \bigcup_{n\in\omega}\Sigma^n$. Then Σ^ω is named, $\Diamond$-saturated, $\exists$-saturated, and maximal consistent. The proof is the same as that given in [1]. We use **Name** to prove the consistency of Σ^0. We use **Paste** to show the consistency of Case 3; this rule gives us exactly what is required. **Paste** is a primitive rule in $\mathsf{K2}$ and a derived rule in both $\mathsf{K1}$ and $\mathsf{K3}$.

Case 4 introduces a small (but significant) deviation from Henkin's recipe for the quantifiers. We don't simply witness the $\exists$ quantifier with a new constant $c_{m,a}$, rather we use the rigidified constant $@_i c_{m,a}$, where i names the world where the existential formula is to be evaluated. Note that $@_i c_{m,a}$, like $c_{m,a}$, is of type a. The change does not effect consistency, and the proof is essentially the standard one; see [1] for details. $\dashv$

Building Hybrid Henkin Structures

We come to the core construction. Recall that a structure $\mathcal{M}$ is a pair of the form $\langle \mathcal{S}, \mathsf{F}\rangle$, where $\mathcal{S} = \langle\langle \mathsf{D}_a\rangle_{a\in\mathsf{TYPES}}, W, R\rangle$ and F a denotation function. So we have two tasks. The first is to define the type hierarchy $\langle \mathsf{D}_a\rangle_{a\in\mathsf{TYPES}}$, the second is to define $\langle W, R\rangle$ and F. Let's deal with the first task.

Definition 21 *Let Δ be a named, $\Diamond$-saturated and $\exists$-saturated maximal consistent set. Then:*

- *For all α_a, $\beta_a \in \mathsf{RIGIDS}_a$: $\alpha_a \approx_\Delta \beta_a$ iff $\alpha_a = \beta_a \in \Delta$, for every $a \in \mathsf{TYPES}-\{t\}$. The* ***rigidity equivalence class*** *of α_a, notation $[\alpha_a]_\Delta$, is the set $\{\beta_a \mid \alpha_a \approx_\Delta \beta_a\}$.*
- *For φ, $\psi \in \mathsf{ME}_t : \varphi \approx_\Delta \psi$ iff $\varphi = \psi \in \Delta$.The* ***truth equivalence class*** *of φ, notation $[\varphi]_\Delta$, is the set $\{\psi \mid \varphi \approx_\Delta \psi\}$.*

When Δ is clear from context we will usually write $\approx$ instead of $\approx_\Delta$, and $[\alpha]$ instead of $[\alpha]_\Delta$. It is straightforward to check that both rigidity equivalence and truth equivalence are equivalence relations.

This leads to the key result: all the equivalence classes needed when building the type hierarchy can be represented by rigidified constants.

Theorem 22 (Rigid Representatives) *Let Δ be a maximal consistent set which is named, $\Diamond$-saturated and $\exists$-saturated.*

1. *Let $h \in \mathsf{HYB}$ and $\alpha \in \mathsf{ME}_t$. Then $[\alpha] = [@_h\bot]$ or $[\alpha] = [@_h\top]$.*
2. *Let $h \in \mathsf{HYB}$ and $\alpha_a \in \mathsf{RIGIDS}_a$, such that $a \neq t$. Then there is a constant $c_a \in \mathsf{CON}$ such that $[\alpha_a] = [@_h c_a]$.*

Proof The proof is by induction on type structure. We give the proof for type t expressions, as state variables are of type t. The case for type e expressions and the inductive step can be found in [1].

Let $h \in \mathsf{HYB}$ and $\alpha \in \mathsf{ME}_t$. Assume that $\alpha \in \Delta$. But $\alpha \to (\alpha = \top) \in \Delta$, by propositional logic. Thus $\alpha = \top \in \Delta$, and $\alpha = @_h\top \in \Delta$, by Axiom 7c and maximal consistency. Hence $[\alpha] = [@_h\top]$. On the other hand, if we assume that $\alpha \notin \Delta$, both $\neg\alpha$ and $\neg\alpha \to (\alpha = \bot) \in \Delta$, and similar reasoning lets us conclude that $[\alpha] = [@_h\bot]$. A remark: since for any $h \in \mathsf{HYB}$ we have that $\vdash @_h\bot = \bot$ and $\vdash @_h\top = \top$, by Axiom 7c $[@_h\bot] = [\bot]$ and $[@_h\top] = [\top]$. So the choice of h is irrelevant; there are only two truth equivalence classes. $\dashv$

Theorem 23 (Hierarchy Theorem) *Given a maximal consistent set Δ, which is named, $\Diamond$-saturated and $\exists$-saturated, there is a family of domains $\langle \mathsf{D}_a\rangle_{a\in\mathsf{TYPES}}$ and a function Φ such that:*

1. *Φ is a bijection from BB (Building Blocks) to $\bigcup_{a\in\mathsf{TYPES}} \mathsf{D}_a$, where*

$$\mathsf{BB} = \bigcup_{a\in\mathsf{TYPES}\setminus\{t\}} \{[\alpha_a] \mid \alpha_a \in \mathsf{RIGIDS}_a\} \cup \{[\varphi] \mid \varphi \in \mathsf{ME}_t\}.$$

2. *$\mathsf{D}_t = \{\Phi([\varphi]) \mid \varphi \in \mathsf{ME}_t\}$ and $\mathsf{D}_a = \{\Phi([\alpha_a]) \mid \alpha_a \in \mathsf{RIGIDS}_a\}$ for $a \neq t$.*

Proof The proof is essentially Henkin's; we refer the reader to [1] for full details. However we will give the case for type t expressions as we want to be explicit about how state variables are handled. We define D_t to be the two elements set $\mathsf{D}_t = \{[@_j\bot], [@_j\top]\}$, and for every $\varphi \in \mathsf{ME}_t$ we define:

$$\Phi([\varphi]) = \begin{cases} [@_j\top] & \text{iff} \quad \varphi \in \Delta \\ [@_j\bot] & \text{iff} \quad \neg\varphi \in \Delta, \end{cases}$$

where the nominal j is arbitrary. It is immediate by the first part of the Rigid Representatives Theorem that Φ is well-defined, one-to-one, and onto. Note that the definition of Φ covers state variables, as they are of type t. We will need to use the definition of $\Phi([s])$ when we prove Theorem 27. $\dashv$

Now for the second task, defining $\langle W, R\rangle$ and F. We first define an equivalence relation over elements of HYB.

Definition 24 *Let Δ be a maximal consistent set. For $h, h' \in \mathsf{HYB}$, we shall define $h \approx^N h'$ to hold if and only if $@_h h' \in \Delta$. Clearly, if $h \approx^N h'$ holds, it means that h and h' name the same world, and it is easy to show that $\approx^N$ is an equivalence relation on HYB. For any $h \in \mathsf{HYB}$, we shall define $[h]^N$ to be $\{h' \in \mathsf{HYB} : h \approx^N h'\}$.*

Definition 25 (Hybrid Henkin Structures) *Let Δ be a maximal consistent set which is named, $\Diamond$-saturated and $\exists$-saturated. The **hybrid Henkin structure** $\mathcal{M} = \langle \mathcal{S}, \mathsf{F} \rangle$ **over** Δ is made up of:*

1. *The skeleton $\mathcal{S} = \langle \langle \mathsf{D}_a \rangle_{a \in \mathsf{TYPES}}, W, R \rangle$, defined by:*
   - *(a) $\langle \mathsf{D}_a \rangle_{a \in \mathsf{TYPES}}$, as given by the Hierarchy Theorem,*
   - *(b) $W = \{[i]^N \mid i \text{ is a nominal}\}$,*
   - *(c) $R = \{\langle [i]^N, [j]^N \rangle \mid @_i \Diamond j \in \Delta\}$.*
2. *F is a function with domain $\mathsf{NOM} \cup \mathsf{CON}$, defined by:*
   - *(a) For $c_{n,a} \in \mathsf{CON}$, $\mathsf{F}(c_{n,a})$ is a function from W to D_a, such that $\mathsf{F}(c_{n,a})([i]^N) = \Phi([@_i c_{n,a}])$.*
   - *(b) For $i \in \mathsf{NOM}$, $\mathsf{F}(i)$ is a function from W to $\mathsf{D}_t = \{[@_i \top], [@_i \bot]\}$, such that $(\mathsf{F}(i))([j]^N) = [@_i \top]$ iff $i \in [j]^N$.*

We need to check that hybrid Henkin structures are indeed well-defined structures, but this is straightforward. Further details can be found in [1].

General Interpretation and Completeness

One last detail remains: defining our variable assignment. The following definition extends the definition in [1] to cover state variables. We remind the reader that Φ is the bijection defined in the proof of the Hierarchy Theorem, and that for any state variable s, $[s]^N$ is the equivalence class of elements of HYB that name the same world as s.

Definition 26 *The **hybrid Henkin assignment** on $\mathcal{M}$ is the function g defined as follows. For every $v_a \in \mathsf{VAR}_a$ we have:*

$$g(v_a) = \Phi([v_a]),$$

and for every state variable $s \in \mathsf{SVAR}$ we have:

$$g(s) = [s]^N.$$

Theorem 27 *Let $\mathcal{M}$ be a hybrid Henkin structure and g its hybrid Henkin assignment. For all meaningful expressions β_b and for all $i \in \mathsf{NOM}$ we have:*

$$[[\beta_b]]^{\mathcal{M}, [i]^N, g} = \Phi([@_i \beta_b]).$$

Proof By induction on the formation of meaningful expressions. A proof covering most steps is given in [1]. Here we give the steps for state variables in formula position and the $\downarrow$ binder.

So suppose that β_b is a state variable s. We want to show, for any nominal i, that $[[s]]^{\mathcal{M},[i]^N,g} = \Phi([@_i s])$. Now, in the hybrid Henkin structure $T = [@_j \top]$ and $F = [@_j \bot]$, where j is an arbitrary nominal. So by the semantic definition for state variables in formula position, the fact that $g(s) = [s]^N$, and Definition 24 we have that:

$$[[s]]^{\mathcal{M},[i]^N,g} = \begin{cases} [@_j \top] & \text{iff} \quad g(s) = [i]^N \quad \text{iff} \quad [s]^N = [i]^N \quad \text{iff} \quad @_i s \in \Delta \\ [@_j \bot] & \text{iff} \quad g(s) \neq [i]^N \quad \text{iff} \quad [s]^N \neq [i]^N \quad \text{iff} \quad @_i s \notin \Delta \end{cases}$$

This immediately gives us the equivalence we require, for from the definition of Φ given in the proof of the Hierarchy Theorem and maximal consistency:

$$\Phi([@_i s]) = \begin{cases} [@_j \top] & \text{iff} \quad @_i s \in \Delta \\ [@_j \bot] & \text{iff} \quad \neg @_i s \in \Delta \quad \text{iff} \quad @_i s \notin \Delta \end{cases}$$

This establishes the case for state variables in formula position.

So suppose β_b is $\downarrow s\varphi$. We now show that $[[\downarrow s\varphi]]^{\mathcal{M},[i]^N,g} = \Phi([@_i \downarrow s\varphi])$. Predictably, this is where we use the DA schema:

$$\begin{array}{rcll} [[\downarrow s\varphi]]^{\mathcal{M},[i]^N,g} & = & [[\varphi]]^{\mathcal{M},[i]^N,g\frac{[i]^N}{s}} & \text{Semantic definition} \\ & = & [[\varphi(\frac{i}{s})]]^{\mathcal{M},[i]^N,g} & \text{Lemma 14} \\ & = & \Phi([@_i(\varphi(\frac{i}{s}))]) & \text{Inductive hypothesis} \\ & = & \Phi([@_i \downarrow s\varphi]) & \mathsf{DA} \end{array}$$

The only other novel case is for expressions of the form $@_s\varphi$, and this is straightforward by the induction hypothesis. $\dashv$

Corollary 28 *A pair $\langle \mathcal{M}, g \rangle$ where $\mathcal{M}$ is a hybrid Henkin structure and g is its Henkin assignment is a general interpretation.*

Proof The induction underlying the proof of the previous theorem shows that every expression has an interpretation in the appropriate domain of the hierarchy, and this is precisely what we require of general interpretations. $\dashv$

Theorem 29 (Henkin theorem) *Every consistent set of meaningful expressions of type t has a general interpretation satisfying it.*

Proof Let Γ be a consistent set of meaningful expressions of type t. By the Lindenbaum lemma there is a maximal consistent extension Δ of Γ which is named, $\Diamond$-saturated and $\exists$-saturated. Because Δ is named, there is a nominal k in Δ. By Theorem 27 and Corollary 28 there is a general interpretation $\langle \mathcal{M}, g \rangle$ such that, for all $\beta_t \in \mathsf{ME}_t$ we have that:

$$[[\beta_t]]^{\mathcal{M},[k]^N,g} = \Phi([@_k \beta_t]).$$

Let $\varphi \in \Gamma$. Hence $@_k\varphi \in \Delta$. But this means that $\Phi([@_k\varphi]) = [@_k\top]$. Hence $[[\varphi]]^{\mathcal{M},[k]^N,g} = [@_k\top]$. That is, φ is true in this model at the world $[k]^N$. $\dashv$

Theorem 30 (Completeness) *For all* $\Gamma \subseteq \mathsf{ME}_t$ *and* $\varphi \in \mathsf{ME}_t$*, the following holds:* $\Gamma \models \varphi$ *implies* $\Gamma \vdash \varphi$

Proof Standard. $\dashv$

5 The bounded fragment

When logicians talk about completeness for first-order logic, they typically talk about *the* completeness theorem, singular, the result first proved by Gödel in his 1929 PhD thesis, and proved in the now standard fashion by Henkin in his celebrated 1949 paper [12]. Modal logicians, on the other hand, typically talk of completeness theorems, plural. Even readers with only passing acquaintance with modal logic may well have heard of modal logics boasting such names as K, T, $\mathsf{S4}$, $\mathsf{S5}$, $\mathsf{S4.3}$ and many many more. Why this difference in perspective?

The reasons are partly historical. Modal logic began as a largely syntactic attempt to pin down such concepts as strict implication, necessity and possibility, and many systems were considered. Although algebraic methods were sometimes used, it was not until the introduction of possible world semantics in the late 1950s, by pioneers such as Kripke, Hintikka, Kanger, that light was shed on this proliferation of logics. For example, it became clear — indeed, easy to see — that $\mathsf{S4}$ is the set of formulas valid on pre-ordered frames $\langle W, R\rangle$, that is, frames where R is reflexive and transitive. Modal completeness results of the early 1960s typically attempted to show that particular axioms of interest had simple semantic characterizations. There were successful attempts to prove more general completeness results, the best known being Sahlqvist's theorem, but such results tended to be complex. And in the early 1970s, incompleteness results were proved in the (seemingly simple) setting of propositional modal logic. All in all, thinking in terms of *logics* is probably the most natural way of coping with this complex landscape of results and partial results.

In first-order logic, thanks to the strength and simplicity of the completeness theorem, the situation is more straightforward. Instead of thinking in terms of different first-order logics, we tend to think in terms of theories. For example, if we want to work with partial orders (that is, antisymmetric pre-orders) we would form the theory consisting of the following sentences:

$$\forall x Rxx \qquad \forall x\forall y(Rxy \land Ryx \rightarrow x = y) \qquad \forall x\forall y\forall z(Rxy \land Ryz \rightarrow Rxz).$$

The first-order completeness theorem guarantees that the consequences of these sentences are precisely the first-order formulas valid on partial orders. Of course, we can, and often do, talk of "the first-order logic of partial orders". Nonetheless, the simplicity and generality of the completeness theorem makes it natural to think in terms of theories, in a way that is uncommon in modal logic.

As its name suggests, hybrid logic lives somewhere in the middle. We shall now prove a completeness result which covers many useful hybrid theories of frame structure. The results discussed below are all standard in (propositional and first-order) hybrid logic; the point of our discussion is to show how straightforwardly they lift to HTT. Indeed, stronger results are known in the hybrid literature, and many of them lift with similar ease to HTT. But we have selected the result below because it is simple and elegant and showcases the locality of the $\downarrow$ binder, a recurrent theme in this paper.

Let's start with an example. Suppose we are working with HTT but wish to work with partially ordered frames. Then we simply form the following theory:

$$\downarrow s(\Diamond s) \qquad \downarrow s\Box(\Diamond s \rightarrow s) \qquad \downarrow s\Box\Box \downarrow t @_s \Diamond t.$$

These sentences pin down reflexivity, antisymmetry, and transitivity respectively. Note that the transitivity sentence makes use of the store and retrieve interplay between $\downarrow$ and @. This sentence says: starting from a point which we have temporally labelled s, if we label t any point that is modally accessible from s in two steps (the $\Box\Box$), then at s we can modally access t in one step (the $\Diamond t$). It is not difficult to see that this formula will be true at *all* the worlds in an HTT model if and only if the frame of the model is transitive. That is, this hybrid sentence defines transitivity.

Another example. If we want to work with models whose frames are *strict* partial orders (that is, irreflexive, asymmetric and transitive) we could work with the following sentential theory:

$$\downarrow s(\Diamond\neg s) \qquad \downarrow s(\Box\neg\Diamond s) \qquad \downarrow s\Box\Box \downarrow t @_s \Diamond t.$$

Again these expressions define the conditions we are interested in. And this leads to the first question that will occupy us: if we add such sentences as extra axioms, do we *automatically* have completeness? Is the completeness result we have proved for HTT like the completeness theorem for first-order logic, in that it supports thinking in terms of theories rather than logics? It is, but to prove this we must first be precise about the axioms we intend to use.

As axioms we allow ***pure nominal-free sentences***. The "pure" means that they contain no constants; "nominal free" and "sentences" are self explanatory. To put it another way, we generate ***pure nominal-free expressions*** as follows. As basic expressions, we only have state variables, and complex expressions can only have the following form:

$$\neg\varphi \mid \varphi \wedge \psi \mid \Box\varphi \mid @_s\varphi \mid \downarrow s\varphi.$$

This is a small fragment of HTT, containing only expressions of type t. But it is also a well-studied fragment of propositional hybrid logic. Don't be fooled by its apparent simplicity: this fragment is undecidable (see [2]) and capable of defining many important frame classes. The axioms we will work with are the *sentences* of this fragment. The above examples are all sentences of this kind (we define the additional booleans and $\Diamond$ in the usual way).

Most of the information in a model $\mathcal{M} = \langle\langle\langle \mathsf{D}_a\rangle_{a\in\mathsf{TYPES}}, W, R\rangle, \mathsf{F}\rangle$ and an assignment g is irrelevant for pure nominal-free expressions. Indeed, all that is relevant is the frame $\langle W, R\rangle$ and the assignment that g makes to the state variables. Thus for expressions φ in this fragment, we are simply looking at interpretations at some world $w \in W$ with respect to an assignment g:

$$[[\varphi]]^{\langle W,R\rangle,w,g}.$$

This leads us to ***frame validity***. We say that a pure nominal-free expression is valid on a frame $\langle W, R\rangle$ if and only if for all $w \in W$, and all assignments g, we have that $[[\varphi]]^{\langle W,R\rangle,w,g} = T$. We write $[[\varphi]]^{\langle W,R\rangle}$ when φ is valid on $\langle W, R\rangle$.

And now for completeness. We want to use pure nominal-free sentences as additional axioms in HTT. And we want to prove a completeness result that tells us: if we add as extra axioms (for example) the hybrid theory of strict partial orders given above, then for any consistent set of type t expressions we can always build a verifying model over a *strictly partially ordered* frame. How can we do so? As we have seen, Theorem 29 will indeed give us a model for consistent sets of expressions, and if we have added extra axioms the hybrid Henkin structure will make them all true too. But that is not enough: we need to show that the axioms are valid on the underlying frame. But for pure nominal-free sentences this is easy.

Lemma 31 *Let φ be a pure nominal-free sentence, let $\langle W, R\rangle$ be a frame, and let g be a fixed but arbitrary assignment on $\langle W, R\rangle$. Suppose that for all $w \in W$ we have $[[\varphi]]^{\langle W,R\rangle,w,g} = T$. Then $[[\varphi]]^{\langle W,R\rangle}$, that is, φ is valid on $\langle W, R\rangle$.*

Proof Suppose that for some assignment g, and all worlds $w \in W$ we have that $[[\varphi]]^{\langle W,R\rangle,w,g} = T$. But φ is a *sentence* so the choice of assignment is irrelevant. That is, at any $w \in W$, for any assignment g' we have that $[[\varphi]]^{\langle W,R\rangle,w,g'} = T$. So φ is valid on $\langle W, R\rangle$ as claimed. $\dashv$

Definition 32 *Let φ be a pure nominal-free sentence. Then $\mathsf{Fr}(\varphi)$ is the class of frames $\langle W, R\rangle$ such that $[[\varphi]]^{\langle W,R\rangle}$. That is, $\mathsf{Fr}(\varphi)$ is the class of all frames that validate φ. A sentential theory Th is a set of pure nominal-free sentences; note that all such sets are countable. Then $\mathsf{Fr}(\mathsf{Th})$ is the class of frames $\langle W, R\rangle$ such that for all $\varphi \in \mathsf{Th}$, $\langle W, R\rangle$ belongs to $\mathsf{Fr}(\varphi)$. That is, $\mathsf{Fr}(\mathsf{Th})$ is the class of frames that validate all sentences in the theory Th.*

*By a **pure nominal-free sentential extension** of our axiomatization(s), we mean the addition as extra axioms of all the pure nominal-free sentences in such a theory Th. A set of type t expressions Γ is Th-consistent if and only it is consistent in this enriched system.*

Theorem 33 (Extended Henkin theorem) *Let Th be a set of pure nominal-free sentences. Every Th-consistent set Γ of meaningful expressions of type t has a general interpretation $\langle\mathcal{M}, g\rangle$ that satisfies it, such that the frame $\langle W, R\rangle$ underlying $\mathcal{M}$ belongs to $\mathsf{Fr}(\mathsf{Th})$.*

Proof For the first claim, we proceed as in the proof of Theorem 29. Let Γ be a Th-consistent set of meaningful expressions of type t. We use our Lindenbaum construction to obtain a maximal consistent extension Δ of Γ which is $\Diamond$-saturated, $\exists$-saturated, and named by some nominal k. By Theorem 27 and Corollary 28 there is a general interpretation $\langle \mathcal{M}, g\rangle$ such that, for all $\beta_t \in \mathsf{ME}_t$ the following holds:

$$[[\beta_t]]^{\mathcal{M},[k]^N,g} = \Phi([@_k\beta_t]).$$

As we saw, this shows that all expressions in Δ (and hence Γ) are true in this general interpretation at the world $[k]^N$. Note that this includes all the sentences in Th, for as Γ is Th-consistent, every sentence in Th will be added to Δ at some stage of the Lindenbaum construction.

But for the second part of theorem, we need to show that all axioms in Th are valid on the frame underlying $\mathcal{M}$, not merely that they are true. We do so as follows. Let $\varphi \in \mathsf{Th}$. As φ is an axiom, then for all nominals i, by @-generalization we have that $\vdash @_i\varphi$. So by maximal consistency, all these expressions belong to Δ. But every world in the hybrid Henkin structure is an equivalence class of nominals. So (here we use the displayed equality again) at every world $[i]^N$ we have that $[[\varphi]]^{\mathcal{M},[i]^N,g} = T$. But φ is a pure nominal-free sentence, so by Lemma 31 it is valid on $\langle W, R\rangle$. But φ was an arbitrary element of Th, so all sentences in Th are valid on $\langle W, R\rangle$. Hence $\langle W, R\rangle$ is in $\mathsf{Fr}(\mathsf{Th})$, which is what we wanted to prove. $\dashv$

What frame classes are covered by this result? Here's a syntactic answer. At the start of this section we gave the three axioms characteristic of partial orders in a first-order language of frames. It was a simple first-order language: a two-place relation R plus the equality symbol. Here we define the bounded fragment of this first-order frame language. We we will use s, t and so on — that is, the symbols we typically use for state variables — as first-order variables. We generate its ***bounded fragment*** as follows:

$$Rst \mid s = t \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists t(Rst \wedge \varphi), \text{where } s \neq t.$$

Now, all pure nominal-free expressions translate into the bounded fragment:

$$\begin{array}{rcl}
ST_s(t) & = & (s = t) \text{ for all state variables } t \\
ST_s(\neg\varphi) & = & \neg ST_s(\varphi) \\
ST_s(\varphi \wedge \psi) & = & ST_s(\varphi) \wedge ST_s(\varphi) \\
ST_s(\Diamond\varphi) & = & \exists t(Rst \wedge ST_t(\varphi)) \\
ST_s(@_t\varphi) & = & ST_t(\varphi) \\
ST_s({\downarrow}t\varphi) & = & (ST_s(\varphi))(\frac{s}{t})
\end{array}$$

In the translation clause for $\downarrow$ we have used $(\frac{s}{t})$ to indicate the substitution of the (first-order) variable s for free occurrences of the (first-order) variable t. This translation uses the relevant clauses of the ***standard translation***, which is widely used in orthodox modal logic and hybrid logic; see [4] or [3] for more detailed accounts. Note that ST_s translates every pure nominal-free *sentence* into a bounded formula with s as its sole free variable. It follows by induction that for all sentences φ, frames $\langle W, R\rangle$ and $w \in W$ that:

$$[[\varphi]]^{\langle W,R\rangle,w} \textit{ iff } \langle W, R\rangle \models ST_s(\varphi)[s \leftarrow w].$$

Here $\langle W, R\rangle \models ST_s(\varphi)[s \leftarrow w]$ means first-order satisfaction, with the unique free variable s in $ST_s(\varphi)$ being assigned w as value. Note that we have written $[[\varphi]]^{\langle W,R\rangle,w}$ rather that $[[\varphi]]^{\langle W,R\rangle,w,g}$ because, as φ is a sentence, the choice of variable assignment is irrelevant. Summing up: all pure nominal-free sentences define frame conditions expressible in the bounded fragment with one free variable s. Hence a pure nominal-free sentence φ is valid on a frame $\langle W, R\rangle$ if and only if $\langle W, R\rangle \models \forall s\varphi$, where $\models$ indicates first-order truth in $\langle W, R\rangle$. So every frame condition expressible by a pure nominal-free sentence can be expressed by a bounded sentence.

And the converse also holds, for we can translate the bounded fragment into the pure nominal-free expression as follows:

$$\begin{array}{rcl}
HT(s = t) & = & @_s t \\
HT(Rst) & = & @_s \Diamond t \\
HT(\neg\varphi) & = & \neg HT(\varphi) \\
HT(\varphi \wedge \psi) & = & HT(\varphi) \wedge HT(\psi) \\
HT(\exists t(Rst \wedge \varphi)) & = & @_s \Diamond {\downarrow} t(HT(\varphi))
\end{array}$$

Again we can show by induction that for every frame $\langle W, R\rangle$, if we translate a bounded first-order formula θ with s as its only free variable, then

$$\langle W, R\rangle \models \theta[s \leftarrow w] \textit{ iff } [[{\downarrow} s HT(\theta)]]^{\langle W,R\rangle,w}.$$

Taken together, these two results show that if we restrict attention to bounded formulas containing at most one free variable, pure nominal-free sentences are a hybrid notation for capturing bounded frame conditions.

The bounded fragment is interesting because it is a *local* fragment of first-logic: it only lets us quantify over accessible entities. Feferman was the first to consider bounded fragments (see [10]); he did so in set theory, using $\in$ rather than R as his accessibility predicate. The link between bounded fragments and hybrid logic was first explored in [2]; more can be said, here we have only noted the bare essentials. For a start, there are elegant semantic characterizations: roughly speaking, pure nominal-free sentences allow us to talk about frame classes that are both closed under and reflect point generated subframes. Moreover, Theorem 33 can be extended to allow free state variables (or nominals) in the axioms. But for further discussion we refer the reader to [9], the authoritative source. Here we shall return to Henkin.

We remarked that Henkin's work fits well with hybrid logic because of the first-order perspective which underlies his best known work. In this section we have been explicit about the first-order character of our hybrid technology. As we have previously remarked, Theorem 33 is atypical in its generality, at least when viewed from orthodox modal logic. But there is no puzzle here: it is the first-order character of key hybrid tools that makes such results possible, and it is the use of Henkin models which makes them easy to prove.

6 It's Henkin, all the way down

In this paper we proved a general completeness theorem for a hybrid type theory called HTT, probably the strongest hybrid logic that has yet been explored. The result built upon earlier work on a system called BHTT, which only used the basic hybrid tools of nominals and the @ operator. The $\downarrow$ binder is the tool that differentiates HTT from the earlier system, and it is $\downarrow$ that allows us to capture the bounded fragment. The influence of Henkin's work throughout this paper should be obvious. For a start, we used Henkins's methods of constants to allow $\Diamond$-prefixed expressions to be witnessed by nominals. Secondly, our used of general interpretations and the construction of the type hierarchy in the completeness proof is (pretty much) one hundred percent Henkin. Thirdly, the first-order nature of pure nominal-free sentences allowed us to extend our K-style completeness result(s) to cover a wide class of hybrid theories, and the proof of the Theorem 33 was straightforward because we were working with a hybrid Henkin structure: as each world is named in such structures, it was easy to verify the validity of the additional axioms.

But to close this paper we want to claim that the most profound impact of Henkin's work on hybrid logic occurs neither at the level of type theory, nor even at the level of first-order hybrid logic, but at the level of *propositional* hybrid logic, and indeed, right down at the level of *basic* hybrid logic, where the only hybrid tools at our disposal are @ and nominals. To put it another way: a recurring theme in the development of hybrid completeness has been that richer languages hitch a free ride on the underlying basic hybrid logic. This is certainly the case for first-order hybrid logic, and the completeness results for HTT and BHTT repeat this pattern.

Let's look closer. Consider first the role of @. For a start, it drives much of the model construction process. Because it lets us rigidify arbitrary type, it lets us state rigidity restrictions on axioms, prove the Rigid Representatives Theorem, and specify an appropriate world for every formula in a suitably saturated maximal consistent set of sentences (these points are made in more detail in [1] where BHTT was proved complete). Moreover, we have seen that two of our axiomatizations, K1 and K2 pretty much reduce $\downarrow$ to a spectator role as far as axiomatics is concerned. The crucial **DA** schema is a local schema: it tells us how to deal with $\downarrow$ at a named world, but we rely on @ to distribute this schema to all named worlds, and the logic of @ is captured in basic hybrid logic. The K3 axiomatization eliminates the non-orthodox basic rules and shows how $\downarrow$ can play a more active role in the axiomatization. But here too we rely on @ to insist that **DA** holds at all named worlds.

And this mention of non-orthodox rules brings us to the heart of our claim. Such rules are not non-orthodox in any interesting sense. Our earlier remarks on the link between the **Name** and **Bounded Generalization** rules and natural deduction hinted at this, but to close this paper we turn to the K2 axiomatization, which uses the basic hybrid logic rule **Paste**. The following remarks draw on and elaborate the discussion on pages 445-447 of [4].

Here is the **Paste** rule:

$$\frac{\vdash @_i\Diamond j \wedge @_j\varphi \rightarrow \theta}{\vdash @_i\Diamond\varphi \rightarrow \theta}\ .$$

As we saw in the proof of our Lindenbaum lemma, this directly licenses the use of witness nominals. And if we read this rule from bottom-to-top we see that it boils down to the following tableau rule:

$$\begin{array}{c} @_i\Diamond\varphi \\ \hline @_i\Diamond j, \quad j \textit{ new} \\ @_j\varphi \end{array}$$

That is: if in the course of a tableaux proof we encounter $@_i\Diamond\varphi$ we are free to decompose this into the near-atomic formula $@_i\Diamond j$ (where j is new) and the simpler $@_j\varphi$. This directly embodies the idea of Henkin-style nominal witnessing, and in one form or another is the basis for most tableau systems for hybrid logic. Deductively, at least, what makes hybrid logic better behaved that orthodox modal logic is the possibility of eliminating $\Diamond$ in this way. Tableau rules for the other connectives are easy to define, dealing with $\Diamond$ is the tricky part. This is the rule that opens to door to usable (non-axiomatic) proof procedures.

But why should we call this rule orthodox? The answer takes us back (yet again) to first-logic and the world of Henkin. Consider the following table. This uses the Standard Translation (recall Section 5) to show the content of the tableau rule, viewed from the perspective of the first-order frame language (enriched with a first-order constant i for every nominal i):

Hybrid Logic	First-Order Frame Language
$@_i\Diamond\varphi$	$\exists s(Ris \wedge ST_s(\varphi))$
	$Rij \wedge ST_j(\varphi)$
$@_i\Diamond j$	Rij
$@_j\varphi$	$ST_j(\varphi)$

The translation of the tableaux rule is a fragment of a first-order tableaux proof. Moreover, it makes clear that (from a first-order perspective) all that is going on is a skolemisation (that is, a witnessing) of the existential quantifier, followed by a conjunction reduction. First-order orthodoxy reigns.

Ultimately, this is why Henkin's work is so important to hybrid logic. Diamond witnessing is the central idea that has driven the development of hybrid deduction and completeness theory. But in hybrid logic, Henkin's use of witnesses is put to work in a simple, decidable propositional language, namely basic hybrid logic. This gives stability and generality to completeness results for basic hybrid logic, and enables them to support completeness in richer hybrid systems (for example, hybrid logics containing the $\downarrow$ binder) and combinations of hybrid logic with first-order logic and type theory. Hybrid logic? It's a suggestive name. But it could without exaggeration be called: Henkin-style modal logic.

Acknowledgements

The authors are grateful to the Spanish Ministerio de Economía y Competitividad for funding the project *Logica Intensional Hibrida* (*Hybrid Intensional Logic*), FFI2013-47126-P, hosted by the Universidad de Salamanca.

References

[1] Carlos Areces, Patrick Blackburn, Antonia Huertas, and María Manzano. Completeness in hybrid type theory. *Journal of Philosophical Logic*, 43:209–238, 2014.

[2] Carlos Areces, Patrick Blackburn, and Maarten Marx. Hybrid logics: Characterization, interpolation and complexity. *Journal of Symbolic Logic*, pages 977–1010, 2001.

[3] Carlos Areces and Balder ten Cate. Hybrid logic. In *Handbook of Modal Logic*, pages 821–868. Elsevier, 2007.

[4] Patrick Blackburn, Maarten De Rijke, and Yde Venema. *Modal Logic*, volume 53. Cambridge University Press, 2002.

[5] Patrick Blackburn and Jerry Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4(3):251–272, 1995.

[6] Patrick Blackburn and Balder ten Cate. Pure extensions, proof rules, and hybrid axiomatics. *Studia Logica*, 84:277–322, 2006.

[7] Patrick Blackburn and Miroslava Tzakova. Hybrid languages and temporal logic. *Logic Journal of the IGPL*, 7(1):27–54, 1999.

[8] Torben Braüner. *Hybrid logic and its proof-theory*, volume 37. Springer, 2010.

[9] Balder ten Cate. *Model theory for extended modal languages*. PhD thesis, University of Amsterdam, 2005. ILLC Dissertation Series DS-2005-01.

[10] Solomon Feferman. Persistent and invariant formulas for outer extensions. *Compositio Mathematica*, 20:29–52, 1968.

[11] George Gargov and Valentin Goranko. Modal logic with names. *Journal of Philosophical Logic*, 22:607–636, 1993.

[12] Leon Henkin. The completeness of the first-order functional calculus. *The Journal of Symbolic Logic*, 14(3):159–166, 1949.

[13] Leon Henkin. Completeness in the theory of types. *The Journal of Symbolic Logic*, 15(2):81–91, 1950.

[14] Leon Henkin. The discovery of my completeness proofs. *The Bulletin of Symbolic Logic*, 2(2):127–158, 1996.

[15] María Manzano. *Extensions of First Order Logic*. Cambridge University Press, 1996.

[16] Johan Van Benthem and Kees Doets. Higher-order logic. In *Handbook of philosophical logic*, pages 275–329. Springer, 1983.