Internalizing Labelled Deduction

Patrick Blackburn

Abstract

This paper shows how to internalize the Kripke satisfaction definition using the *basic hybrid language*, and explores the proof theoretic consequences of doing so. As we shall see, the basic hybrid language enables us to transfer classic Gabbay-style labelled deduction methods from the metalanguage to the object language, and to handle labelling discipline logically. This internalized approach to labelled deduction links neatly with the Gabbay-style rules now widely used in modal Hilbert-systems, enables completeness results for a wide range of first-order definable frame classes to be obtained automatically, and extends to many richer languages. The paper discusses related work by Jerry Seligman and Miroslava Tzakova and concludes with some reflections on the status of labelling in modal logic.

1 Introduction

Modern modal logic revolves around the Kripke satisfaction relation:

$$\mathcal{M}, w \Vdash \varphi.$$

This says that the model $\mathcal{M}$ satisfies (or forces, or supports) the modal formula φ at the state w in $\mathcal{M}$. Now, the $\Vdash$ relation is inductively defined in the metalanguage. In this paper we investigate the proof-theoretic consequences of *internalizing* this relation; that is, mirroring it in the object language.

We internalize $\Vdash$ by defining the *basic hybrid language*. This is obtained by making two changes to ordinary propositional modal logic. First we add a new sort of atomic symbol called *nominals*. These symbols, which we typically write as i, j, k and l, are ordinary formulas. Syntactically they behave just like the familiar propositional variables; they can be negated, conjoined, placed in the scope of modalities, and so on. But there is a crucial *semantic* difference: any nominal will be true at *exactly one* state in any model. To add nominals to a modal language is in effect to add a collection of *labels*; each nominal 'labels' the unique state it is true at.

Second, we shall allow ourselves to form a new kind of formula: given a nominal i, and any formula φ, we can form $i:\varphi$. These are the formulas that mirror the $\Vdash$ relation; a formula of the form $i:\varphi$ will be true at any state in a model if and only if φ is true at the (unique) state that i labels. We call these new $i:$ prefixes *satisfaction operators*.

There are many reasons for being interested in hybrid languages, but this paper is mostly concerned with *deduction*. So: what does the basic hybrid language have to offer modal proof theory? Here's a preliminary list.

Internalizes the labelling mechanism The basic hybrid language *obviously* internalizes the apparatus of modal labelled deduction. Modal labelled deduction (which traces back to Fitting [20] and has been proposed as a systematic approach to deduction by Dov Gabbay; see in particular [21]) makes use of labels in the metalanguage to regulate the proof process. In such approaches, a modal formula φ can be prefixed by a metalinguistic label l; a common notation for this is $l:\varphi$, which means that φ is true at the state labelled l. The extended modal languages considered in this paper treat such expressions as members in good standing of the object language.

Internalizes *discipline* on labels This is perhaps the single most important point to grasp. "Why?", the reader may be wondering, "treat nominals as formulas? Why not simply add the $i:\varphi$ construct, and only allow nominals to occur in such contexts?"

There are many reasons for treating nominals as formulas, but the most relevant for present purposes is this: as Gabbay has repeatedly emphasized, it is *not* enough to think of labelled deduction simply in terms of labels. Labels are the building blocks — but using labels in proofs requires that we have a *discipline* on labels; that is, a systematic algebra of label manipulation.

By treating nominals as formulas that can be recursively combined with $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\Diamond$, $\Box$ and $:$ *we fully internalize labelling discipline.* Discipline won't be imposed from the outside, it will emerge from the semantics of our language. To put it another way, when labels and satisfaction are internalized in the object language, *discipline* is simply another word for *logic*.

Link with Gabbay-style rules Internalized labelled deduction provides a transparent link between labelled deduction methods and the Gabbay-style proof rules that have played an increasingly important role in modal Hilbert systems over the last twenty years (see [22]). As I shall show, in the basic hybrid language the basic Gabbay-style proof rule simply *is* the crucial sequent rule for internalized labelled deduction. Indeed, this observation was the starting point of the paper.

Generality Internalized labelled deduction is general in several senses. Once the minimal calculus has been specified, calculi for a wide range of stronger logics can be obtained more-or-less automatically. This includes complete calculi not only for frame classes that ordinary modal languages can define (such as the class of transitive frames) but for many classes that ordinary modal languages cannot define (such as the class of irreflexive frames). Moreover, internalized labelled deduction extends to many other modal languages, and indeed to more expressive hybrid languages in which we can *bind* labels.

Ease of use Because of the presence of nominals and satisfaction operators, we will be able to make use of *standard* proof methods; no additional metalinguistic machinery will be needed. I shall formulate internalized labelled deduction as both an unsigned tableau system and as a sequent calculus. Familiarity with standard classical proof methods should enable the reader to carry out proofs in either system without difficulty.

Internalized labelled deduction does not originate in this paper: it traces back to Jerry Seligman's work on deduction methods for Situation Theory (see [41, 42, 43]). In more recent work, Seligman (see [44]) has investigated what I call *mixed* systems, an idea that has been independently explored by Miroslava Tzakova (see [48]). Mixed methods are an interesting alternative to full internalization, and I shall discuss them in Section 8. Two other recent approaches are worth noting: Demri's sequent systems for nominal tense logic (see [17]), and the Demri and Goré display calculi (see [18]) for nominal tense logic enriched with the difference operator. These papers don't make direct use of satisfaction operators, but many of the underlying ideas are similar to those discussed here. Also relevant is Hasle's [29] HOL implementation of Prior's third grade tense logic.

2 Internalizing Kripke semantics

Let's start by recalling the syntax and semantics of ordinary propositional modal logic. Given a set of *propositional symbols* PROP $= \{p, q, r, \ldots\}$, the set of formulas of the *basic modal language* (over PROP) is defined as follows:

$$\text{WFF} := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \Diamond\varphi \mid \Box\varphi.$$

The only defined connective I shall use is $\leftrightarrow$, and this has its usual definition.

The basic modal language is interpreted on *models*. A model $\mathcal{M}$ is a triple (W, R, V). Here W is a non-empty set of states, and R is a binary relation on W. The pair (W, R) is called the *frame* underlying $\mathcal{M}$, and $\mathcal{M}$ is said to be a model over (W, R). Sometimes additional properties may be demanded of R (for example transitivity, irreflexivity, density, or antisymmetry) and I shall discuss such demands later; but for the time being R is simply an arbitrary binary relation on W. V (the *valuation*) is a function with domain PROP and range $Pow(W)$; it tells us at which states (if any) each propositional symbol is true.

Interpretation is carried out using the *Kripke satisfaction definition*. This is defined as follows. Let $\mathcal{M} = (W, R, V)$ and $w \in W$. Then:

$$\begin{array}{lll}
\mathcal{M}, w \Vdash p & \text{iff} & w \in V(p), \text{ where } p \in \text{PROP} \\
\mathcal{M}, w \Vdash \neg\varphi & \text{iff} & \mathcal{M}, w \nVdash \varphi \\
\mathcal{M}, w \Vdash \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \Vdash \varphi \text{ and } \mathcal{M}, w \Vdash \psi \\
\mathcal{M}, w \Vdash \varphi \vee \psi & \text{iff} & \mathcal{M}, w \Vdash \varphi \text{ or } \mathcal{M}, w \Vdash \psi \\
\mathcal{M}, w \Vdash \varphi \rightarrow \psi & \text{iff} & \mathcal{M}, w \nVdash \varphi \text{ or } \mathcal{M}, w \Vdash \psi \\
\mathcal{M}, w \Vdash \Diamond\varphi & \text{iff} & \exists w'(wRw' \ \& \ \mathcal{M}, w' \Vdash \varphi) \\
\mathcal{M}, w \Vdash \Box\varphi & \text{iff} & \forall w'(wRw' \ \Rightarrow \ \mathcal{M}, w' \Vdash \varphi).
\end{array}$$

If $\mathcal{M}, w \Vdash \varphi$ we say that φ is *satisfied* in $\mathcal{M}$ at w.

So far, everything should be familiar — so let's go a step further and *internalize* the Kripke satisfaction definition by enriching the basic modal language with *nominals* and *satisfaction operators*. Let NOM be a nonempty set, disjoint from PROP. The elements of NOM are called *nominals*, and we typically write them as i, j, k and l. We now define the *basic hybrid language* (over PROP and NOM) to be the following collection of formulas:

$$\text{WFF} := i \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \to \psi \mid \Diamond\varphi \mid \Box\varphi \mid i\!:\!\varphi.$$

For any nominal i, we shall call the symbol sequence $i:$ a *satisfaction operator*.[1] A formula of the form $i:\varphi$ or $\neg i:\varphi$ is called a *satisfaction statement*; such statements form the backbone of our proof systems. Note that $\neg i:\varphi$ is $\neg(i:\varphi)$. It *cannot* be $(\neg i):\varphi$, for this expression is not a formula.

The key point to note is that *nominals are perfectly ordinary formulas.* Although nominals will label states (that is, although they will play much the same semantic role as first-order constants), syntactically they are simply a second sort of atomic formula. For example, the previous definition permits us to form wffs like

$$\Diamond(i \wedge p) \wedge \Diamond(i \wedge q) \to \Diamond(p \wedge q)$$

in which labelling information (encoded by nominals) is freely mixed with arbitrary propositional information using the booleans and modalities. This mixing of information is vital to the proof theoretical work of the paper: it is *precisely* because labelling has been fully internalized into the object language that the discipline required to control deduction will emerge. For example, under the semantics defined below, the previous formula is *valid*: because the antecedent guarantees the existence of both a successor state labelled i at which p is true, and a successor state labelled i at which q is true, then there must be at least one successor state (namely the unique state labelled i) at which *both* p and q are true. In short, our semantics will support a rich logic of labelling, and this will carry much of the deductive load for us.

So let's be precise about the intended semantics. We have already said that we want nominals to function as labels by being true at exactly one state, and that we want $i:\varphi$ to be an object language correlate of the metalinguistic relation $w \Vdash \varphi$. It is straightforward to achieve these goals. Given a frame $\mathcal{F} = (W, R)$, a valuation on $\mathcal{F}$ will now be a function with domain PROP$\cup$NOM and range $Pow(W)$. And now for the key concept: a *standard* valuation on $\mathcal{F}$ is a valuation such that for all $i \in$ NOM, $V(i)$ is a *singleton* subset of W. That is, a standard valuation makes each nominal true at a *unique* point in a model; the nominal labels this point by being true there and nowhere else. We call the unique point w that belongs to $V(i)$ the *denotation* of i under V. We say that a model $\mathcal{M} = (W, R, V)$ is standard if and only if V is standard, and we interpret the basic hybrid language on standard models by adding the following two clauses to the Kripke satisfaction definition:

$\mathcal{M}, w \Vdash i$	iff	w is the denotation of i under V
$\mathcal{M}, w \Vdash i:\varphi$	iff	$\mathcal{M}, \mathrm{i} \Vdash \varphi$, where i is the denotation of i under V.

That is, to evaluate $i:\varphi$ we jump to the denotation of i and evaluate φ there. Note that if a satisfaction statement (whether of the form $i:\varphi$ or $\neg i:\varphi$) is satisfied at *one* state in a model, then it is satisfied at *all* states in that model.

If φ is satisfied at all states in a standard model $\mathcal{M}$, then we say that φ is *globally satisfied* in $\mathcal{M}$ and write $\mathcal{M} \Vdash \varphi$. If $\mathcal{F}$ is a frame, and for all *standard* models $\mathcal{M}$ based on $\mathcal{F}$ we have $\mathcal{M} \Vdash \varphi$, then we say that φ is *valid* on $\mathcal{F}$ and write $\mathcal{F} \Vdash \varphi$. If φ

[1] Other suggestive terms include: *jump* operator, *situating* operator, *localizing* operator, and *supports* operator. Another term is *at* operator, and $i:\varphi$ is written $@_i\varphi$ in [8, 11, 12, 2].

is valid on all frames we call it a validity and write $\Vdash \varphi$. These concepts are extended to sets of formulas in the obvious way.

It should now be clear that the basic hybrid language internalizes the Kripke satisfaction definition; arguably, it does so in a particularly natural way. The basic hybrid language is very much a *modal* language. Although we have introduced labels into the object language, these are simply special atomic formulas; they involve no strange new syntactic mechanisms. What about the satisfaction operators? These are also modally natural, for they are simply normal modal operators. To see this, observe that for every nominal i,

$$\Vdash i\!:\!(\varphi \to \psi) \to (i\!:\!\varphi \to i\!:\!\psi).$$

That is, all satisfaction operators validate the modal normality schema. Moreover, we have that

$$\Vdash \varphi \quad \text{implies} \quad \Vdash i\!:\!\varphi.$$

That is, the rule of necessitation holds for every satisfaction operator too. In short, each satisfaction operator supports the minimal modal logic K, and is thus a normal modal operator. Actually, satisfaction operators support a far richer logic than K. For example, each prefix validates the self-duality schema,

$$\Vdash i\!:\!\varphi \leftrightarrow \neg i\!:\!\neg\varphi,$$

and a lot more besides; we shall see more examples later.

Moreover, the basic hybrid language turns out to be a *simple* modal language; in fact, its satisfiability problem (deciding whether a given formula is satisfiable) is no harder than the satisfiability problem for ordinary propositional modal logic:

Theorem 1 The satisfiability problem for the basic hybrid language is PSPACE-complete.

Proof PSPACE-hardness is clear, for the basic hybrid language contains the satisfiability problem for ordinary propositional modal logic as a special case, and this problem is known to be PSPACE-hard (see [34]). A PSPACE algorithm for basic hybrid satisfiability is given in [2].

So from a complexity-theoretic perspective, the basic hybrid language is offering us something for nothing — but what exactly? Essentially this: the power to construct a *modal theory of state equality* and a *modal theory of state succession* in the object language. To see why, note that $i\!:\!j$ says that the states labelled i and j are identical, while the statement $\neg i\!:\!j$ says they are distinct. That is, these near-atomic satisfaction statements are analogous to first-order atomic formulas of the form $c = d$ and $\neg(c = d)$. And we can also form state succession statements: $i : \Diamond j$ states that the state labelled j is a successor of the state labelled i, while $\neg i\!:\!\Diamond j$ denies this. Again, these are near-atomic satisfaction statements, and they are clearly analogous to first-order atomic formulas of the form Rcd and $\neg Rcd$. The proof methods discussed in this paper systematically exploit such near-atomic satisfaction statements.[2]

[2]Model theoretically, these remarks could be summed up as follows: the basic hybrid language

The basic hybrid language is modally natural for yet another reason: the *pure formulas* it contains offer a simple handle on another key modal theme: frame definability and its interaction with completeness. A pure formula is a formula that contains no propositional variables. That is, the only atomic symbols pure formulas contain are nominals. As for frame definability, a formula φ *defines* a class of frames F if and only if $\mathcal{F} \Vdash \varphi$ iff $\mathcal{F} \in \mathsf{F}$ (that is, a formula defines the class of frames it is valid on). When we say a modal formula defines a certain property (for example, transitivity) we mean it defines the class of all frames with that property.

What properties can we define using pure formulas? For a start, we can define many properties that are definable in the basic modal language:

$i \to \Diamond i$	*Reflexivity*
$i \to \Box\Diamond i$	*Symmetry*
$\Diamond\Diamond i \to \Diamond i$	*Transitivity*
$\Diamond i \to \Diamond\Diamond i$	*Density*
$\Diamond i \wedge \Diamond j \to \Diamond(i \wedge j)$	*Determinism.*

More interestingly, we can define many properties that are *not* definable in the basic modal language:

$i \to \neg\Diamond i$	*Irreflexivity*
$i \to \Box\neg\Diamond i$	*Asymmetry*
$i \to \Box(\Diamond i \to i)$	*Antisymmetry*
$\Diamond\Diamond i \to \neg\Diamond i$	*Intransitivity*
$\Diamond i$	*Universality.*

(*Universality* means $\forall x \forall y Rxy$.) Note that the above examples make no use of satisfaction operators. With their help we can define even more:

$j\!:\!\Diamond i \vee j\!:\!i \vee i\!:\!\Diamond j$	*Trichotomy*
$i\!:\!(\neg j \wedge \neg k) \to j\!:\!k$	*At Most 2 States.*

(*Trichotomy* means $\forall x \forall y(Rxy \vee x = y \vee Ryx)$.) Note that the formula defining *At Most 2 States* is an encoding of the Pigeonhole Principle that can be generalized to any natural number n. For example, to pick out the frames containing at most 3 states we write:

$$i\!:\!(\neg j \wedge \neg k \wedge \neg l) \wedge j\!:\!(\neg k \wedge \neg l) \to k\!:\!l.$$

All the properties just defined are first-order. This is no coincidence. It is straightforward to extend the Standard Translation of the basic modal language into classical logic to the basic hybrid language (various ways of doing so can be found in [8] and [2]) and it follows immediately from the form these extensions take that pure formulas can only define first-order properties. The key point is that nominals can be

enables us to define *diagrams*. We do so as follows. If $\mathcal{M} = (W, R, V)$ is a model, add a new nominal i_w to NOM for every $w \in W$, and extend V to a new valuation V^W by defining $V^W(i_w) = \{w\}$. Define $\mathcal{M}^W$ to be (W, R, V^W). We then define the positive diagram of $\mathcal{M}$ to be the set of all satisfaction statements of the form $i\!:\!j$, $i\!:\!p$, and $i\!:\!\Diamond j$ (in the extended language) that are satisfied in $\mathcal{M}^W$, and define the diagram of $\mathcal{M}$ to be the set of all such satisfaction statements, together with their negations, that are satisfied in $\mathcal{M}^W$. As its name suggests, the diagram of a model condenses an exact picture of the model into a set of near-atomic satisfaction statements. In fact, the proof methods we shall develop can be thought of, in a very real sense, as *diagrammatic reasoning*.

translated to free first-order variables (or indeed, first-order constants). Hence, as no ordinary propositional variables appear in pure formulas, no second-order quantification is need to account for their frame-defining powers.[3] A closely related (and for present purposes, more important) point about pure formulas is the following: not only do they define the stated properties, they *automatically* give rise to complete calculi for the class of frames with that property. This is remarkably easy to prove, as we shall see when we turn to extended completeness results in Section 7.

Some general remarks. Not only is the $i:$ symbol likely to be familiar from its metalinguistic role in labelled deduction, the reader has probably seen similar object-level "jump to the named state and evaluate" operators elsewhere. For example Prior [38] makes use of a $T(s, \varphi)$ construct in his *second grade tense logic* (this says, φ is true at the time s), the *Situation Calculus* of McCarthy and Hayes [35] makes use of the same idea, and such operators are characteristic of the *Topological Logic* of Rescher and Urquhart [40]. Moreover Allen [1] introduces a $\mathrm{Holds}(s, \varphi)$ operator for temporal representation in AI (this says that φ holds at the interval named s), Cresswell [16] makes use of satisfaction operators for applications in natural language semantics, van Benthem [4] introduces a $\mathtt{goto}_s$ operator, and Buvač, Buvač and Mason [14] have explored similar ideas to define logics of context. And I doubt if this is even close to being an exhaustive list: satisfaction operators are appealing and natural, and have probably been invented on many other occasions.

However, while all this work is related to the ideas of this paper these systems differ from the hybrid languages in a fundamental way. Although they incorporate labels into the object language,

they don't treat labels as first-class citizens.

That is, they don't treat labels as *formulas*: we are not allowed to negate them, conjoin them, prefix them with modalities, and so on. In these languages we are only allowed to use nominals to form satisfaction operators. The first person to take labels seriously — and to realize that being serious meant viewing labels as special *propositions* — was probably Arthur Prior, the founder of tense logic. Prior introduced nominals (see Chapter V, section 6, and Appendix B, section 3, of [38]) to define what he called *third grade tense logic*. Third grade tense logic is the tool that enabled Prior to develop what he regarded as an adequate perspective on temporal logic, and was probably the first hybrid language to be developed.

Prior's motivation was largely philosophical, and a full discussion of his work would take us too far afield. For present purposes it is enough to note that Prior's philosophical journey leads to interesting *technical* territory. As is clear from our frame

[3] There's a lot more that could be said here, and no space to say it all, but the following remarks may be helpful. First, an immediate consequence is that pure formulas cannot define all the properties that the basic modal language can, for the use of propositional variables enables *second-order* properties to be defined. To give a standard example, the Löb axiom $\Box(\Box p \to p) \to \Box p$ defines the property of being transitive and converse well founded, which is not first-order definable. If we replace the propositional variable p in Löb by the nominal i we obtain the pure formula $\Box(\Box i \to i) \to \Box i$ which defines a simple *first-order* condition; any frame which validates the Löb axiom will have this first-order property, but not conversely. Second, although pure formulas can define first-order properties of frames that the basic modal language cannot define, the reverse is also true. For example, the Church-Rosser property is definable in the basic modal language by $\Diamond\Box p \to \Box\Diamond p$, but no pure formula defines this property. For further information on frame definability, see [23], [6], [8] and [2].

definability examples, treating labels as formulas has important consequences for expressivity. We have not yet begun to bind labels (many of the formalisms mentioned above allow this) but we are already seeing gains in our ability to define frame classes. Treating labels as first-class citizens gives rise to a fine-grained expressivity hierarchy, but if we use labels only in satisfaction operators, we lose much of this.[4] And more is at stake than expressivity: the ability to freely manipulate labels in the object language is the key idea needed to define general proof systems. And we *don't* need to jump all the way up to formalisms with full first-order expressive strength to reap this benefit (many of the formalisms mentioned above are first-order equivalent); the core deductive machinery required is already present in basic hybrid language. That is, we have not only located an expressivity hierarchy, we have located a *deductive* hierarchy too.

Here are some pointers to the literature on hybrid languages. For satisfaction operators in the presence of nominals or bindable nominals see [41, 42, 43, 44, 8, 11, 12, 48, 2]. There are also papers on hybrid languages containing nominals but no satisfaction operators: see in particular [37, 24, 25, 23, 6, 17, 18]; some of these papers use the universal modality (or the more powerful difference operator) to simulate the effect of satisfaction operators, something Prior himself did.[5] Finally, there is a long tradition of work which explores hybrid languages in which it is possible to bind nominals using the quantifiers $\forall$ and $\exists$: see [38, 13, 39, 36, 37, 26, 27, 28, 8, 9, 10].

3 Internalizing labelled deduction

We are now ready to internalize labelled deduction. I first do so using a Smullyan-style unsigned tableau system (see [46]), which I will then reformulate as a sequent calculus.

I doubt if it's possible to beat unsigned tableau systems for sheer simplicity.[6] To prove φ we test whether $\neg\varphi$ is a contradiction, and we do so by building a tree with $\neg\varphi$ at its root. The tableau expansion rules tell us how to break this initial formula down and expand the tree using the pieces. In the propositional case the expansion process is mechanical (albeit non-deterministic), and the meaning of the expansion rules is transparent (they simply restate the satisfaction conditions). The method uses virtually no metatheoretic apparatus: a proof (that is, a closed tableau) is simply a

[4] If we did not treat nominals as formulas — that is, if we only allowed them to occur as the first argument of satisfaction operators — then there would be a dramatic drop in our ability to define classes of frames. For example, irreflexivity would no longer be definable.

[5] The universal modality A has as satisfaction definition: $\mathcal{M}, w \Vdash A\varphi$ iff $\mathcal{M}, w' \Vdash \varphi$ for *all* states w' in $\mathcal{M}$. That is, whereas $i:$ internalizes the notion of *satisfaction* at a point, A internalizes the notion of *global satisfaction*. Using the universal modality we can define satisfaction operators (define $i{:}\varphi$ to be $A(i \rightarrow \varphi)$) but the universal modality cannot be defined in the basic hybrid language.

While interesting and important, languages containing both A and nominals have drawbacks. If we jump straight to such languages we obscure the fact that the expressivity and deductive gains begin lower in the hierarchy. Moreover, adding A gives us an EXPTIME-complete satisfiability problem even if we don't add nominals. Finally, once we start allowing ourselves to bind labels, adding A catapults us straight up to full first-order strength, even if we only make use of the intrinsically local $\downarrow$ binder discussed later in this paper. For further discussion see [8, 11, 12].

[6] For example, Hodges makes use of them in [30], an introduction to logic for readers with no formal background. The book succeeds in this task remarkably well, helped by the simplicity of the tableau calculi.

tree whose nodes are decorated with object language formulas. There are no signs, metalinguistic labels, or indeed anything else.

The key ideas of unsigned tableau systems adapt straightforwardly to the basic hybrid language. Suppose we want to see if a formula φ is valid. Now, if φ is *not* valid, it is possible to build a model for its negation at some state in some model. So, let i be a nominal that does not occur in φ (i acts as a label for the putative falsifying state). We systematically search for a model for $\neg i:\varphi$; if we *can't* find one, φ is valid. We search for models by building a tree with $\neg i:\varphi$ at its root. A proof will simply be a tree whose nodes are decorated with object language formulas.

The tree will be expanded in accordance with the following rules. These fall into four groups: rules for the booleans, rules for the satisfaction operators, rules for $\Diamond$ and $\Box$, and rules for coping with the modal theories of state equality and state succession. In what follows, φ and ψ are metavariables over arbitrary wffs, s, t, and u are metavariables over nominals, and a is a metavariable over *new* nominals (that is, nominals that have not been used so far in that branch of the tableau).

First, the rules for the booleans:

$$\frac{s:\neg\varphi}{\neg s:\varphi}\,[\neg] \qquad\qquad \frac{\neg s:\neg\varphi}{s:\varphi}\,[\neg\neg]$$

$$\frac{s:(\varphi\wedge\psi)}{\begin{array}{c}s:\varphi\\ s:\psi\end{array}}\,[\wedge] \qquad\qquad \frac{\neg s:(\varphi\wedge\psi)}{\neg s:\varphi \mid \neg s:\psi}\,[\neg\wedge]$$

$$\frac{s:(\varphi\vee\psi)}{s:\varphi \mid s:\psi}\,[\vee] \qquad\qquad \frac{\neg s:(\varphi\vee\psi)}{\begin{array}{c}\neg s:\varphi\\ \neg s:\psi\end{array}}\,[\neg\vee]$$

$$\frac{s:(\varphi\rightarrow\psi)}{\neg s:\varphi \mid s:\psi}\,[\rightarrow] \qquad\qquad \frac{\neg s:(\varphi\rightarrow\psi)}{\begin{array}{c}s:\varphi\\ \neg s:\psi\end{array}}\,[\neg\rightarrow]$$

These rules should be self-explanatory: they merely state what it means for the formula above the horizontal line (the input to the rule) to be satisfied at the point labelled by a nominal s in a model. The $\neg\wedge$-rule, the $\vee$-rule, and the $\rightarrow$-rule are called *branching rules* for they yield two alternative outputs.

Now for the satisfaction operators:

$$\frac{s:t:\varphi}{t:\varphi}\,[:] \qquad\qquad \frac{\neg s:t:\varphi}{\neg t:\varphi}\,[\neg:]$$

That is, given a pair of nested satisfaction operators, the outermost can be discarded.

Now for the most interesting rules; those dealing with the modalities:

$$\frac{s:\Diamond\varphi}{\begin{array}{c}s:\Diamond a\\ a:\varphi\end{array}}\,[\Diamond] \qquad\qquad \frac{\neg s:\Diamond\varphi \quad s:\Diamond t}{\neg t:\varphi}\,[\neg\Diamond]$$

$$\frac{s:\Box\varphi \quad s:\Diamond t}{t:\varphi}\,[\Box] \qquad\qquad \frac{\neg s:\Box\varphi}{\begin{array}{c}s:\Diamond a\\ \neg a:\varphi\end{array}}\,[\neg\Box]$$

Note that the $\Diamond$-rule and $\neg\Box$-rules are stated using the metavariable a. This means that when we apply these rules to some formula in a tableau, we should choose a nominal that hasn't been used so far in the tableau construction process. The $\Diamond$-rule is subject to the following side condition: we cannot apply it to formulas of the form $s{:}\Diamond\varphi$ where φ is a nominal. Let's examine these rules more closely.

The $\Diamond$-rule and the $\neg\Box$-rule are called *existential rules*. These are the only rules that introduce new nominals into a tableau. Why do they do this? Consider the $\Diamond$-rule. This decomposes the existential demand made by formulas of the form $s{:}\Diamond\varphi$ into two subdemands: (1) that there is a successor state to the state labelled s (we invent a brand new name a for this successor), and (2) that the state labelled a satisfies φ. The $\neg\Box$-rule works analogously. In short, just as with all the other rules, we have expressed the satisfaction conditions using the resources available in the object language.

Some comments. First, there is a clear analogy between these rules and the way existential quantifiers are handled in tableau systems for first-order logic. Second, there is an even more obvious analogy with the way Gabbay handles the logic of $\Diamond$ in labelled deduction systems (see [21]):

$$\frac{s{:}\Diamond\varphi}{\textit{create } a,\ sRa,\ \textit{and } a{:}\varphi}$$

In fact, the only real difference is that Gabbay proceeds by manipulating labels *metalinguistically* (in effect, he makes use of a programming language containing expressions such as '*create*', '*and*', 'R', ':', and a supply of labels, to manipulate object language formulas) whereas here we work with an object language rich enough to support the required deduction step internally. Third, it should now be clear why the $\Diamond$-rule carries a side condition: if we already have the information $s:\Diamond j$, then it is pointless to generate $s:\Diamond a$ and $a:j$. Doing so would simply invent a new label for an already labelled state.[7] Fourth, it should also be clear that we are in a genuine sense doing *diagrammatic reasoning* (see Footnote 3). All tableau systems are based on model building. Here we search for models by trying to build the diagrams that describes them.

Let's turn to the $\Box$-rule and the $\neg\Diamond$-rule, the *universal rules*. Both rules are binary (they take *two* formulas as input) and the second input to each rule (the minor premiss) is a near-atomic satisfaction statement of the form $s:\Diamond t$. They should be read as follows: if a branch contains a pair of formulas of the form shown above the line, then we can extend that branch by adding the formula shown below the line. I have written the two input formulas side-by-side to indicate that there is no ordering intended on the input (that is, if you can find a matching pair of formulas anywhere on the branch, no matter which occurs first, you can apply the rule). In essence, these rules treat formulas of the form $s{:}\Box\varphi$ and $s{:}\neg\Diamond\varphi$ as constraints on successors of the state labelled s. Such formulas can be regarded as demons scanning tableau branches, demons summoned by formulas of the form $s{:}\Diamond t$.

Although we have given a rule for each connective, we are not yet finished. One of the key intuitions about the basic hybrid language is that it internalizes the mecha-

[7] Note that this restriction does not apply to the $\neg\Box$ rule. For example, if we have the information that $\neg s{:}\Box j$, then we should be able to name the j falsifying successor, and by using $\neg\Box$ rule we can do so: we obtain $s{:}\Diamond a$ and $\neg a{:}j$.

nisms needed to formulate modal *theories* of state equality and state succession. But nothing in the system so far gets to grips with these (crucial) theories. How are we to do so? In fact, there are many interesting possibilities here, but one natural response is to bolt on a rewrite system:

$$\frac{[s \text{ on branch}]}{s\!:\!s}\;\textsf{[Ref]} \qquad \frac{t\!:\!s}{s\!:\!t}\;\textsf{[Sym]} \qquad \frac{s\!:\!t \quad t\!:\!\varphi}{s\!:\!\varphi}\;\textsf{[Nom]} \qquad \frac{s\!:\!\Diamond t \quad t\!:\!u}{s\!:\!\Diamond u}\;\textsf{[Bridge]}$$

The meaning of the Ref and Sym rules should be clear (by "*s on branch*" in the statement of the Ref rule I simply mean that some formula on the branch in question contains an occurrence of s). What about the Nom rule? This reflects the fact that identical states carry identical information: if on a branch we have the information that $s\!:\!t$ (thus s and t label identical states), and we also have the information that $t:\varphi$, then we should be able to conclude that $s:\varphi$, and this is precisely what Nom lets us do. Note that the "missing" transitivity rule (that is, from $s:t$ and $t:u$ to deduce $s\!:\!u$) is merely the special case of Nom in which φ is u. In short, the Ref, Sym, and Nom rules reflect the fact that the basic hybrid language is strong enough to talk about state equality. Finally, Bridge regulates information about state *succession*.

It's time for some examples. Let's first prove the modal distribution axiom:

$$\Box(p \to q) \to (\Box p \to \Box q).$$

We do so by choosing a nominal that does not occur in this formula (let's choose i), prefixing the formula with $\neg i\!:$, and applying tableau rules:

1	$\neg i\!:\!(\Box(p \to q) \to (\Box p \to \Box q))$		
2	$i\!:\!\Box(p \to q)$		$1, \neg\!\to$
$2'$	$\neg i\!:\!(\Box p \to \Box q)$		Ditto
3	$i\!:\!\Box p$		$2', \neg\!\to$
$3'$	$\neg i\!:\!\Box q$		Ditto
4	$i\!:\!\Diamond j$		$3', \neg\Box, j$
$4'$	$\neg j\!:\!q$		Ditto
5	$j\!:\!p$		$3, 4, \Box$
6	$j\!:\!(p \to q)$		$2, 4, \Box$
7	$\neg j\!:\!p$	$j\!:\!q$	$6, \to$
	✠ $5, 7$ ✠	✠ $4', 7$ ✠	

The annotations in the far right column tell us which rule was applied and where. At line 4 we also indicate which new nominal was introduced by the existential rule $\neg\Box$ (in this case, j). The annotations ✠ $5, 7$ ✠ and ✠ $4', 7$ ✠ signal branch closure (branches containing contradictory information) and say where the conflicting items can be found.

Now, the previous example showed how to use the tableau system to prove an ordinary modal formula; and in fact, it can prove *any* validity of the basic modal language. But it can do a lot more than that. As we have already seen, the basic hybrid language supports a rich logic of labelling. Here's a simple example. The following schema is valid:

$$i\!:\!j \to (j\!:\!\varphi \to i\!:\!\varphi).$$

This schema is essentially a reflection in the object language of the idea captured by the Nom rule. (In fact, this schema plays a key role in the Hilbert system presented in [11].) Any instance of this schema can be proved in seven steps as follows (here k is some nominal that does not occur in φ):

1	$\neg k:(i:j \to (j:\varphi \to i:\varphi))$	
2	$k:i:j$	$1, \neg\!\to$
2′	$\neg k:(j:\varphi \to i:\varphi)$	Ditto
3	$i:j$	$2, :$
4	$k:j:\varphi$	$2', \neg\!\to$
4′	$\neg k:i:\varphi$	Ditto
5	$j:\varphi$	$4, :$
6	$\neg i:\varphi$	$4', \neg:$
7	$i:\varphi$	$3, 5$ Nom
	✠ 6, 7 ✠	

In a similar vein, observe that all instances of $\Diamond i \wedge i:\varphi \to \Diamond\varphi$ are valid. Any instance can be proved as follows (here j does not occur in φ):

1	$\neg j:(\Diamond i \wedge i:\varphi \to \Diamond\varphi)$	
2	$j:(\Diamond i \wedge i:\varphi)$	$1, \neg\!\to$
2′	$\neg j:\Diamond\varphi$	Ditto
3	$j:\Diamond i$	$2', \wedge$
3′	$j:i:\varphi$	Ditto
4	$i:\varphi$	$3', :$
5	$\neg i:\varphi$	$2', 3, \neg\Diamond$
	✠ 4, 5 ✠	

Finally, here's how to prove $i:i$, the simplest validity in the language:

1	$\neg j:i:i$	
2	$\neg i:i$	$1, \neg:$
3	$i:i$	Ref
	✠ 2, 3 ✠	

So we have an easy-to-use minimal calculus — indeed, in terms of simplicity, it lies between Smullyan's propositional and first-order systems, and from the perspective of modal correspondence theory, that's just the way it should be. If desired, the system can be converted into a Smullyan-style *signed* system (see [46]) by adding the metalinguistic symbols $\mathcal{T}$ for true and $\mathcal{F}$ for false. Using this notation, the signed rules for $\Diamond$ are:

$$\frac{\mathcal{T}s:\Diamond\varphi}{\begin{array}{c}\mathcal{T}s:\Diamond a\\ \mathcal{T}a:\varphi\end{array}} \qquad\qquad \frac{\mathcal{F}s:\Diamond\varphi \quad \mathcal{T}s:\Diamond t}{\mathcal{F}t:\varphi}$$

But the use of signs is unnecessary: the basic hybrid language is rich enough to take care of everything.

Let's reformulate the tableau system as a cut-free sequent calculus. In what follows, Γ and Σ are multisets of satisfaction statements and $\longrightarrow$ is the sequent arrow. A sequent is an axiom if it is of the form $\theta \longrightarrow \theta$, where θ is any satisfiability statement. As structural rules we take *Weakening* and *Contraction*.

$$\frac{\Gamma \longrightarrow \Sigma}{\theta, \Gamma \longrightarrow \Sigma}\;\textsf{[WL]} \qquad \frac{\Gamma \longrightarrow \Sigma}{\Gamma \longrightarrow \Sigma, \theta}\;\textsf{[WR]}$$

$$\frac{\theta, \theta, \Gamma \longrightarrow \Sigma}{\theta, \Gamma \longrightarrow \Sigma}\;\textsf{[CL]} \qquad \frac{\Gamma \longrightarrow \Sigma, \theta, \theta}{\Gamma \longrightarrow \Sigma, \theta}\;\textsf{[CR]}$$

Here's a selection of sequent rules for the connectives:

$$\frac{\Gamma \longrightarrow \Sigma, s{:}\varphi}{s{:}\neg\varphi, \Gamma \longrightarrow \Sigma}\;\textsf{[}\neg\textsf{L]} \qquad \frac{s{:}\varphi, \Gamma \longrightarrow \Sigma}{\Gamma \longrightarrow \Sigma, s{:}\neg\varphi}\;\textsf{[}\neg\textsf{R]}$$

$$\frac{s{:}\varphi, s{:}\psi, \Gamma \longrightarrow \Sigma}{s{:}(\varphi \wedge \psi), \Gamma \longrightarrow \Sigma}\;\textsf{[}\wedge\textsf{L]} \qquad \frac{\Gamma \longrightarrow \Sigma, s{:}\varphi \qquad \Gamma \longrightarrow \Sigma, s{:}\psi}{\Gamma \longrightarrow \Sigma, s{:}(\varphi \wedge \psi)}\;\textsf{[}\wedge\textsf{R]}$$

$$\frac{t{:}\varphi, \Gamma \longrightarrow \Sigma}{s{:}t{:}\varphi, \Gamma \longrightarrow \Sigma}\;\textsf{[:L]} \qquad \frac{\Gamma \longrightarrow \Sigma, t{:}\varphi}{\Gamma \longrightarrow \Sigma, s{:}t{:}\varphi}\;\textsf{[:R]}$$

$$\frac{s{:}\Diamond a, a{:}\varphi, \Gamma \longrightarrow \Sigma}{s{:}\Diamond\varphi, \Gamma \longrightarrow \Sigma}\;\textsf{[}\Diamond\textsf{L]} \qquad \frac{\Gamma \longrightarrow \Sigma, t{:}\varphi}{s{:}\Diamond t, \Gamma \longrightarrow \Sigma, s{:}\Diamond\varphi}\;\textsf{[}\Diamond\textsf{R]}$$

$$\frac{t{:}\varphi, \Gamma \longrightarrow \Sigma}{s{:}\Box\varphi, s{:}\Diamond t, \Gamma \longrightarrow \Sigma}\;\textsf{[}\Box\textsf{L]} \qquad \frac{s{:}\Diamond a, \Gamma \longrightarrow \Sigma, a{:}\varphi}{\Gamma \longrightarrow \Sigma, s{:}\Box\varphi}\;\textsf{[}\Box\textsf{R]}$$

If these rules are read from top to bottom, the correspondence between the sequent and tableau rules should be obvious. In particular, note that $\Diamond L$ and $\Box R$ introduce a new nominal a, just as the $\Diamond$-rule and the $\neg\Box$-rule do. But note: as formulated above, the $\Diamond L$ and $\Box R$ rules rely on the presence of contraction. For example, the $\Box L$ rule throws away the copy of $s{:}\Box\varphi$ (and $s{:}\Diamond t$) on the left. But such formulas are essentially *constraints*, and we may need to reuse them later in the proof. Thus we will typically need to apply contraction to duplicate $s{:}\Box\varphi$ before we apply this rule (like the other rules, contraction should be read bottom to top). In a sense, contraction is the sequent counterpart of the tableau system's branch-scanning demons. The contraction rules can be dropped if we absorb their effect into the relevant rules. For example, $\Box L$ could be stated as follows:

$$\frac{t{:}\varphi, s{:}\Box\varphi, s{:}\Diamond t, \Gamma \longrightarrow \Sigma}{s{:}\Box\varphi, s{:}\Diamond t, \Gamma \longrightarrow \Sigma}\;[\Box L]$$

This localizes the effects of contraction, and has more of the flavor of the corresponding tableau rule.

Here are the sequent analogs of Ref, Sym, Nom and Bridge:

$$\frac{s{:}s, \Gamma \longrightarrow \Sigma}{\Gamma \longrightarrow \Sigma} \qquad \frac{s{:}t, \Gamma \longrightarrow \Sigma}{t{:}s, \Gamma \longrightarrow \Sigma} \qquad \frac{s{:}\varphi, \Gamma \longrightarrow \Sigma}{s{:}t, t{:}\varphi, \Gamma \longrightarrow \Sigma} \qquad \frac{s{:}\Diamond u, \Gamma \longrightarrow \Sigma}{s{:}\Diamond t, t{:}u, \Gamma \longrightarrow \Sigma}$$

It is possible to formulate the rewrite system in this comparatively simple way because of the contraction rules lurking in the background. As with the corresponding tableau rule, use of Ref can be restricted to nominals that have occurred at some earlier stage of the proof.

We won't use sequent-based internalized deduction often, though we will return to it in Section 8 when we discuss the work of Jerry Seligman. However the sequent-based presentation makes it easy to link internalized labelled deduction with another aspect of Gabbay's work, so let's turn to this right away.

4 Gabbay-style rules

Nearly twenty years ago, Dov Gabbay augmented the standard Hilbert axiomatization of modal logic with a new kind of proof rule (see [22]). His idea has proved influential (and controversial) and a wide range of similar rules (now known as *Gabbay-style rules* or *rules for the undefinable*) have been developed for many modal languages.

Such rules have one thing in common: they rest on the idea of labelling states. Now, usually no special syntax is provided for these state labels; rather, labelling is simulated using ordinary propositional variables.[8] But of course, in the basic hybrid language we *do* have labels in the object language: so let's formulate a Gabbay-style rule using nominals, and play with it a little.

To keep the first formulation of the rule simple, I shall use the following notation: formulas of the form $\Diamond(s \wedge \varphi)$ will be abbreviated to $\Diamond_s\varphi$. Then a Gabbay-style rule for the basic hybrid language is:

$$\frac{\vdash \Diamond_t \cdots \Diamond_s \Diamond_a \varphi \rightarrow \sigma}{\vdash \Diamond_t \cdots \Diamond_s \Diamond \varphi \rightarrow \sigma}\ \textsf{[Gabbay]}$$

(Here a is a nominal distinct from $s, \cdots, t$ that does not occur in φ or σ.)

This rule is genuinely useful. Enriching axiomatic systems for the basic hybrid language with all instances of this rule enables us to build special models in which each point is labelled by a nominal, which makes it easy to prove a host of strong completeness results (see Lemma 12 below and the accompanying discussion). Unfortunately the rule is rather complex: even in its abbreviated form it is more complex than necessitation (from $\vdash \varphi$ conclude $\vdash \Box\varphi$) the standard modal proof rule. Moreover, we need all instances of this rule, thus we are introducing rules whose "active part" — that is, the $\Diamond_a\varphi$ occurrence — is embedded under arbitrarily deep nestings of modalities. Furthermore, at first glance, the rule may be rather hard to understand. Read contrapositively (from bottom to top) with σ instantiated to a contradiction, it says: if $\Diamond_s \cdots \Diamond_t \Diamond\varphi$ is consistent, then introducing a new name a for the state where φ is true is a consistency preserving operation. This is rather different from the familiar proof rules, and some modal logicians regard Gabbay-style rules as technical tricks that should be used only as a last resort.

But appearances are misleading: the rule has a natural interpretation, and is eminently usable — in fact, once we've made one key simplification we will see that it is nothing but the crucial $\Diamond L$ sequent calculus rule, lightly disguised.

Here's the simplification. The unpleasant aspect of the previous rule is that it requires us to paste in the new nominal a under arbitrarily deep embeddings of diamonds. But, as was pointed out in [11] and [12], we can avoid this with the help of satisfaction operators. In fact, with their help we need only paste in a at a $\Diamond$-depth of 1. Here's the crucial *Paste-1* rule which allows us to do this:

$$\frac{\vdash s{:}\Diamond_a \varphi \rightarrow \sigma}{\vdash s{:}\Diamond \varphi \rightarrow \sigma}\ \textsf{[Paste]}$$

That is, the satisfaction operators enable us to *directly* specify what happens at s (instead of laboriously chaining our way through from t to s) and to paste in the required new successor nominal a. This collapses the stack of nested diamonds.

[8] One exception to this is the work of the Sofia school who make use of a rule called COV in modal languages containing nominals; for further discussion of COV see [36], [23], and [9].

And now $\Diamond L$ rule is staring us in the face. First, let's get rid of of the $\vdash$ symbols, turn the material implication arrow $\rightarrow$ into the sequent arrow $\longrightarrow$, and expand our $\Diamond_a$ abbreviation. This yields:

$$\frac{s\!:\!\Diamond(a \wedge \varphi) \longrightarrow \sigma}{s\!:\!\Diamond\varphi \longrightarrow \sigma}$$

Decomposing the conjunction into a pair of formulas yields:

$$\frac{s\!:\!\Diamond a, a\!:\!\varphi \longrightarrow \sigma}{s\!:\!\Diamond\varphi \longrightarrow \sigma}$$

This rule works in arbitrary deductive contexts, so let's add a lefthand context Γ, and turn σ into a righthand context Σ, thus obtaining $\Diamond L$:

$$\frac{s\!:\!\Diamond a, a\!:\!\varphi, \Gamma \longrightarrow \Sigma}{s\!:\!\Diamond\varphi, \Gamma \longrightarrow \Sigma}\ [\Diamond\mathsf{L}]$$

Far from being strange, or unnatural, or difficult to use, Gabbay style rules are the key to simple and usable proof systems — at least, for logicians prepared to use hybrid languages.

5 Soundness and systematic construction

I turn now to soundness and completeness results for internalized labelled deduction; I shall work with the unsigned tableau presentation. This brief section lays the groundwork for the completeness results that follow; its main purpose is to define a notion of systematic tableau construction general enough to establish strong completeness results for countable languages.

Recall that every formula in a tableau is of the form $s\!:\!\varphi$ or $\neg s\!:\!\varphi$ and that such formulas are called satisfaction statements. Suppose that Σ is a set of satisfaction statements, and that R is one of our tableau rules. Then:

1. If R is *not* a branching rule, and R takes a single formula as input, and Σ^+ is the set obtained by adding to Σ all the formulas (there are at most two) yielded by applying R to $\sigma_1 \in \Sigma$, then we say that Σ^+ is a result of expanding Σ by R.

2. If R is a binary rule (that is, it is either the $\Box$-rule, the $\neg\Diamond$-rule, Nom or Bridge), and Σ^+ is the set obtained by adding to Σ the formula yielded by applying R to $\sigma_1, \sigma_2 \in \Sigma$, then we say that Σ^+ is a result of expanding Σ by R.

3. If R is a branching rule (that is, either the $\neg\wedge$-rule, the $\vee$-rule, or the $\rightarrow$-rule), and Σ^+ is a set obtained by adding to Σ the formula yielded by one of the two possible outcomes of applying R to $\sigma_1 \in \Sigma$, then we say that Σ^+ is a result of expanding Σ by R.

4. If a nominal s belongs to some formula in Σ, then $\Sigma \cup \{s : s\}$ is the result of expanding Σ by Ref.

Definition 2 (Satisfiable by label) Suppose that Σ is a set of satisfaction statements and that $\mathcal{M} = (W, R, V)$ is a standard model. We say that Σ is *satisfied by label* in $\mathcal{M}$ if and only if for all formulas in Σ:

1. If $s:\varphi \in \Sigma$, then $\mathcal{M}, \mathrm{s} \Vdash \varphi$; and

2. if $\neg s:\varphi \in \Sigma$ then $\mathcal{M}, \mathrm{s} \nVdash \varphi$.

(Here s is the denotation of s under V.) We say that Σ is *satisfiable by label* if and only if there is a standard model in which it is satisfied by label.

Lemma 3 (Soundness) Suppose Σ is a set of satisfaction statements that is satisfiable by label. Then, for any rule R, at least one of the sets obtainable by expanding Σ by R is satisfiable by label.

Proof The only non-trivial cases are when R is an existential rule. But even these are straightforward, as the reader can easily check.

Thus our tableau rules cannot lead us astray. If the formula $\neg s:\varphi$ at the root of the tree is satisfiable, then trivially it is satisfiable by label. But then the preceding lemma guarantees that all the formulas on at least one branch of the tableau will by satisfiable by label too.

It is time to turn to *systematic* tableau construction. I shall define a notion of systematicity general enough to prove *strong* completeness for countable languages (languages in which both PROP and NOM are countable sets). That is, ultimately I want to show that any consistent set of formulas in a countable language has a model, not just any finite set of formulas.

Now, the basic idea should be clear. Suppose PROP and NOM are both countable, and let Σ be a set of formulas in the basic hybrid language over these sets. We should pick a nominal i that does not occur in any of these formulas, prefix each of these formulas by $i:$, and start applying tableau rules.[9]

But there's a problem: maybe every nominal in NOM already occurs somewhere in Σ. And anyway, when we apply existential rules, we shall need to have a supply of new nominals at our disposal. So let's take care of this right away. Let NEWNOM be a countable set that is pairwise disjoint with NOM and PROP. Assume that NEWNOM has been enumerated. We shall use its elements as the new nominals needed in the systematic construction. Suppose that i is the first nominal in the enumeration of NEWNOM. Let $\Sigma^i = \{i:\sigma \mid \sigma \in \Sigma\}$. Enumerate the elements of Σ^i and proceed as follows:

> *Stage 1.* Draw a tree consisting of a single node decorated by the first element of Σ^i. Call this T_1. Obviously T_1 is a finite tree.
>
> *Stage* $n+1$. Let T_n be the finite tree constructed at stage n. We now apply all rules that are applicable to formulas (or pairs of formulas) in T_n. As T_n contains only finitely many nodes, only finitely many such applications are possible, hence as no rule returns more than two formulas, the result will be a new finite tree. (As for Ref, we'll assume that we apply it once for each distinct nominal on each branch.) Now, we don't really care in which order the rules are applied, but the following stipulations *are* important:

[9] We prefix by $i:$ rather than $\neg i:$ because we are thinking in terms of consistency rather than provability. The systematic tableau construction defined below is very much an abstract *model building* process. As we shall see, the heart of the strong completeness proof (Theorem 11) is to recover finite proof trees from the infinite trees systematic construction gives rise to.

1. When we apply a non-branching rule, we add the formulas output by the rule to the end of every branch containing the input formula (or pair of input formulas).
2. When we apply a branching rule, we split the end of every branch containing the input formula, and add one possible output to one branch, and the other possible output to the other.
3. When we apply an existential rule, we always use the next unused nominal in NEWNOM as the new nominal we require. (There will always be such a nominal, for we can only have used up finitely many by stage n, and NEWNOM is infinite.)

Once we have applied all the rules, add the $n+1$-th element of Σ^i to the end of every branch. Call the resulting tree T_{n+1}. Clearly T_{n+1} is finite.

The result of this process is an ω-sequence of finite trees, each of which is isomorphically embedded in all its successors. Let T be the tree obtained as the limit of this sequence; T will embody all the information in its predecessors, and will enable us to prove strong completeness in the following section.

6 Hintikka sets and completeness

With the notion of systematic tableau construction defined, we are ready to prove completeness. The proof rests on the notion of Hintikka sets:

Definition 4 (Hintikka Sets) A set of satisfaction statements H is called a *Hintikka set* iff it satisfies the following conditions:

1. For all atoms α, and all nominals s, if $s:\alpha \in H$ then $\neg s:\alpha \notin H$.
2. For all nominals s and t, if $s:\Diamond t \in H$ then $\neg s:\Diamond t \notin H$.
3. If a nominal s occurs in any formula in H, then $s:s \in H$.
4. If H contains a formula that one of the branching rules can be applied to, then it contains at least one of the formulas obtainable by making this application.
5. If H contains a pair of formulas that one of the binary rules can be applied to, then it contains all the formulas obtainable by making this application.
6. If H contains a formula that one of the existential rules can be applied to, then for some nominal i it also contains the formulas that would be obtained by applying that rule to that formula using i as the new nominal a. (We shall call such a nominal i a *witness*.)
7. For any other rule, if H contains a formula that one of the rules applies to, then it contains all the formulas obtainable by making this application.

Items 1, 2 and 3, which regulate what happens to atomic or near-atomic formulas, are among the most crucial demands in the definition: essentially they allow us to fix the diagram of the model we shall eventually build.

Definition 5 Let H be a Hintikka set. Define $Nom(H)$ to be

$$\{i \mid i \text{ is a nominal that occurs in some formula in } H\},$$

and define a binary relation $\sim_H$ on $Nom(H)$ by $i \sim_H j$ iff $i\!:\!j \in H$. Clearly $\sim_H$ is an equivalence relation: item 3 in the definition of Hintikka sets ensures reflexivity, while closure under Sym and Nom guarantees symmetry and transitivity. If $k \in Nom(H)$, then $|k|$ is the equivalence class of k under $\sim_H$.

Lemma 6 Let H be a Hintikka set, and suppose that $i, k \in Nom(H)$. Then the following two assertions are equivalent:

1. $i \sim_H k$
2. For every formula φ, $i\!:\!\varphi \in H$ iff $k\!:\!\varphi \in H$

Proof First suppose that $i \sim_H k$. Then $i\!:\!k \in H$, and hence $k\!:\!i \in H$ too. But then item 2 holds because Hintikka sets are closed under Nom.

For the converse, suppose that $i\!:\!\varphi \in H$ iff $k\!:\!\varphi \in H$. As k occurs in H, we have that $k\!:\!k \in H$. Hence, taking φ to be k, it follows that $i\!:\!k \in H$, which means that $i \sim_H k$.

Definition 7 (Induced models) Given a Hintikka set H, let $\mathcal{H} = (W^H, R^H, V^H)$ be any triple that satisfies the following condition:

1. $W^H = \{|k| \mid k \in Nom(H)\}$.
2. $|i|R^H|j|$ iff $i\!:\!\Diamond j \in H$.
3. $V^H(\alpha) = \{|k| \mid k\!:\!\alpha \in H\}$, for all atoms that occur in H. For any propositional variable p not occurring in H, $V^H(p)$ can be any subset of W^H, while for any nominal i not occurring in H, $V^H(i)$ can be any singleton subset of W^H.

Anticipating the next lemma, we call such $\mathcal{H}$ the *standard models induced by H*.

Lemma 8 Let H be a Hintikka set. Then any triple $\mathcal{H} = (W^H, R^H, V^H)$ of the kind just described is a standard model.

Proof To show that R^H is well-defined, we need to show that for all $i, j, k, l \in Nom(H)$, if $i \sim_H k$ and $j \sim_H l$, then $|i|R^H|j|$ implies that $|k|R^H|l|$. So suppose $i \sim_H k$ and $j \sim_H l$. Further suppose that $|i|R^H|j|$. By definition this means that $i\!:\!\Diamond j \in H$, hence as $i \sim_H k$ by Lemma 6 we have that $k\!:\!\Diamond j \in H$. But as $j \sim_H l$, we also have that $j\!:\!l \in H$. Hence, by Bridge, $k\!:\!\Diamond l \in H$, and thus $|k|R^H|l|$ as required.

It is an immediate consequence of Lemma 6 that V^H is well-defined. But we also need to show that V^H is a *standard* valuation: that is, for all nominals i, $V^H(i)$ is a singleton. By definition this holds for any nominals *not* occurring in H, so suppose i occurs in H. Then $V^H(i)$ contains $|i|$, for by item 3 in the definition of Hintikka sets, $i\!:\!i \in H$. So suppose $|j| \in V^H(i)$. But this means that $j\!:\!i \in H$, which means that $j \sim_H i$, which means that $|j| = |i|$. Thus, for all nominals i, $V^H(i)$ is a singleton subset of W^H as required.

Theorem 9 Let H be a Hintikka set and $\mathcal{H}$ a standard model induced by H. Then:

1. If $i : \varphi \in H$, then $\mathcal{H}, |i| \Vdash \varphi$.

2. If $\neg i : \varphi \in H$, then $\mathcal{H}, |i| \nVdash \varphi$.

That is, every formula in H is satisfied by label in $\mathcal{H}$.

Proof By induction on the number of connectives in φ. If φ is an atomic formula the result is clear. So suppose φ has the form $\Diamond j$, for some nominal j. If $i : \Diamond j \in H$, then by the definition of R^H we have $|i| R^H |j|$. But by the case for atomic formulas, $\mathcal{H}, |j| \Vdash j$, hence $\mathcal{H}, |i| \Vdash \Diamond j$, as required. On the other hand, suppose that $\neg i : \Diamond j \in H$. By item 2 in the definition of Hintikka sets this means that $i : \Diamond j \notin H$, which means that it is *not* the case that $|i| R^H |j|$. Now, by the atomic case we know that $\mathcal{H}, |j| \Vdash j$. Moreover, because V^H is a *standard* valuation we know that $|j|$ is the *only* state where j is true. Hence $\mathcal{H}, |i| \nVdash \Diamond j$.

The inductive steps for booleans and satisfaction operators are straightforward, so let's turns to the modalities. First suppose that φ has the form $\Diamond \psi$. We have just proved the result for the case when ψ is a nominal, so suppose that ψ is some other kind of formula. If $i : \Diamond \psi \in H$, then as ψ is not a nominal it can be used as the input to the $\Diamond$ rule, and so by item 6 in the definition of Hintikka sets there is some witness j such that $i : \Diamond j \in H$ and $j : \psi \in H$. As $i : \Diamond j \in H$, we have that $|i| R^H |j|$; and as $j : \psi \in H$, we have that $\mathcal{H}, |j| \Vdash \psi$ by the inductive hypothesis. Hence $\mathcal{H}, |i| \Vdash \Diamond \psi$ as required.

So suppose instead that $\neg i : \Diamond \psi \in H$. We need to show that for all $|k|$ such that $|i| R^H |k|$, $\mathcal{H}, |k| \nVdash \psi$. So suppose that $|i| R^H |k|$; that is, $i : \Diamond k \in H$. Applying the binary rule $\neg\Diamond$ to $\neg i : \Diamond \psi$ and $i : \Diamond k$ yields $\neg k : \psi$, hence as H is a Hintikka set, by item 5 $\neg k : \psi \in H$. By the inductive step for satisfaction statements we thus have that $\mathcal{H}, |k| \nVdash \psi$, hence as $|k|$ was an arbitrary successor of $|i|$ it follows that $\mathcal{H}, |i| \nVdash \Diamond \psi$ as required.

The argument for formulas of the form $\Box \psi$ is much the same, hence the stated result holds by induction.

Definition 10 (Consistency and provability) A branch of a tableau is closed iff it contains some satisfaction statement and its negation, and a branch which is not closed is open. A tableau is closed iff all its branches are closed. A formula σ is provable iff there is a *finite* closed tableau whose root node is $\neg i : \sigma$ (here i can be any nominal not occurring in σ) and σ is consistent iff $\neg \sigma$ is not provable. A set of formulas Σ is consistent iff for any finite subset Σ^f of Σ, the conjunction of all the formulas in Σ^f is consistent.

Theorem 11 (Strong completeness) Any consistent set of formulas in a countable language is satisfiable in a countable standard model.

Proof Suppose Σ is a consistent set of formulas. Carry out the systematic tableau construction described in the previous section. Suppose for the sake of a contradiction that the tableau T yielded by this process is closed. Let B be any branch on this tableau. As B is closed, it contains some satisfaction statement and its negation. Let

m be the least finite stage of the systematic construction at which these formulas were first present together on (an initial segment of) branch B; define *Discard*(B) to be

$$\{t \mid t \text{ is a node on } B \text{ that does not belong to } T_m\}.$$

(That is, we keep all the information on the branch up to and including the contradiction, and throw away everything generated later.) Define *Discard* to be the union of $\bigcup_B$ *Discard*(B), where B ranges over all branches in T. Let *Clip*(T) be the tree obtained by removing all the nodes in *Discard*. It should be clear that *Clip*(T) is a tree, but is it a *finite* tree? It must be. For suppose it is infinite. Then as every node in T is the mother of at most two nodes, so is every node in *Clip*(T). Hence, by Königs Lemma, *Clip*(T) must contain an infinite branch. But this is impossible: we clipped off all branches finitely many steps away from the root.

Now for the heart of the argument: we shall see that T cannot be closed as we initially assumed. Since *Clip*(T) is a finite tree, all of its nodes are contained in T_n for some finite n. Moreover, by construction, every branch in T_n is an extension of a branch in *Clip*(T), which means that T_n is a *closed* tableau. As we shall now see, this means that

$$\neg(\sigma_1 \wedge \cdots \wedge \sigma_n)$$

is provable, where $\sigma_1, \ldots, \sigma_n$ are the first n formulas in the enumeration of Σ used by the systematic construction. In fact we can read off the required proof from T_n.

We do this as follows. Form a tree with $\neg i : \neg(\sigma_1 \wedge \cdots \wedge \sigma_n)$ at its root, apply the $\neg\neg$-rule to obtain $i : (\sigma_1 \wedge \cdots \wedge \sigma_n)$, and then apply $\wedge$-rule $n-1$ times. Now mimic the sequence of rule applications used to systematically generate T_n. This can be done, because the initial segment of the new tableau contains not only $i : \sigma_1$ but also all the $i : \sigma_2, \ldots, i : \sigma_n$ added in the first n steps of the systematic construction. But mimicking the sequence of rule applications means that we will generate a contradiction on every branch, which means we can prove $\neg(\sigma_1 \wedge \cdots \wedge \sigma_n)$, contradicting our assumption that Σ is consistent. We conclude that T must contain at least one open branch B.

The rest of the proof is standard. Let B be the set of all formulas on B; because of the systematic nature of the tableau construction, B must be a Hintikka set. Let $\mathcal{B}$ be the model induced by B; note that $\mathcal{B}$ is countable as it is built out of equivalence classes of nominals. By the previous theorem, every formula in B is satisfied by label in $\mathcal{B}$. In particular, every formula in Σ^i is satisfied at $|i|$ as required.

7 Extensions

In this section I show how internalized labelled deduction copes with three different kinds of extension. First, I show that the method can be adapted to obtain logics for a wide range of first-order definable frame classes. Second, I show how to adapt it to different modal languages by examining the case of Priorean tense logic. Lastly, I adapt it to stronger hybrid languages in which we can bind labels.

Logics of other frame classes

I gave several examples of properties definable using pure formulas (that is, formulas containing no propositional variables) in Section 2. Some of these properties (for example, transitivity) were definable in the basic modal language, others (for example, irreflexivity) were not. I am now going to let pure formulas be used as axioms in tableau proofs; this will enable us to prove a strong completeness theorem covering many important classes of frames.

Let's start with an example: a tableau system for deterministic frames (that is, frames (W, R) where R is a partial function). Here's the required axiom: $\Diamond i \to \Box i$. This defines determinism and we can use it in tableau proofs as follows. Let j and k be any nominals (not necessarily distinct) that occur on the same branch of a tableau. Then at any stage of the tableau construction we are allowed to add $j:\rho$ to the end of this branch, where ρ is the axiom or results from the axiom by uniformly substituting k for i.[10]

What can we prove in the resulting system? Well, any deterministic frame validates $(\Diamond p \wedge \Diamond q) \to \Diamond(p \wedge q)$, so this should be provable, and indeed it is:

$$
\begin{array}{lccl}
1 & & \neg i:((\Diamond p \wedge \Diamond q) \to \Diamond(p \wedge q)) & \\
2 & & i:(\Diamond p \wedge \Diamond q) & 1, \neg\to \\
2' & & \neg i:\Diamond(p \wedge q) & \text{Ditto} \\
3 & & i:\Diamond p & 2, \wedge \\
3' & & i:\Diamond q & \text{Ditto} \\
4 & & i:\Diamond j & 3, \Diamond, j \\
4' & & j:p & \text{Ditto} \\
5 & & i:\Diamond k & 3', \Diamond, k \\
5' & & k:q & \text{Ditto} \\
6 & & i:(\Diamond j \to \Box j) & \text{Axiom} \\
7 & \neg i:\Diamond j & \mid \quad i:\Box j & 6, \to \\
8 & ✠\ 4,\ 7\ ✠ & k:j & 5, 7, \Box \\
9 & & j:k & 8, \text{Sym} \\
10 & & j:q & 5', 9, \text{Nom} \\
11 & & \neg j:(p \wedge q) & 2', 4, \neg\Diamond \\
12 & & \neg j:p \quad \mid \quad \neg j:q & 11, \neg\wedge \\
 & & ✠\ 4',\ 12\ ✠ \qquad ✠\ 10,\ 12\ ✠ &
\end{array}
$$

That's the basic idea. Let's generalize it and prove an extended strong completeness result.[11] As my starting point, I'll take the following lemma from Gargov and Goranko [23]:

Lemma 12 Let S be a non-empty set of nominals and let `Pure` be a set of pure formulas that is closed under uniform substitution of nominals in S for nominals. (That is, if $\rho \in$ `Pure`, and $s(\rho)$ is obtainable from ρ by uniformly replacing nominals in ρ by nominals from S, then $s(\rho) \in$ `Pure`.) Let $\mathcal{M} = (\mathcal{F}, V)$ be a model such that

[10] We've seen this mechanism before. In essence, this is the way the Ref rule works: the presence of a nominal on a branch allows us to introduce a certain formula built out of that nominal. Given that Ref is there to help us code a modal *theory* of state equality, and that we are now trying to code a modal *theory* of determinism, this similarity is unsurprising.

[11] While the generality of the following completeness result may come as a surprise if you have never worked with nominals before, the only real novelty is that it is being shown for a non Hilbert-style proof system. Historically, the good behavior of pure formulas in Hilbert-style systems has been an important motivation for introducing nominals, and a variety of analogous results have been proved for Hilbert systems in various hybrid languages; see [13], [37], [25], [23], [11] and [12].

$\texttt{Pure}$ is globally satisfied in $\mathcal{M}$, and every state in $\mathcal{F}$ is the denotation of some nominal in S under V. Then $\mathcal{F} \Vdash \texttt{Pure}$.

Proof Suppose for the sake of a contradiction that $\mathcal{F} \nVdash \texttt{Pure}$. Then for some standard valuation V', some state w, and some $\rho \in \texttt{Pure}$ we have that $(\mathcal{F}, V'), w \nVdash \rho$. Let $i_1, \ldots, i_n$ be the nominals occurring in ρ. Let $j_1, \ldots, j_n$ be nominals in S such that $V(j_1) = V'(i_1), \ldots, V(j_n) = V'(i_n)$; such nominals exist, for V labels every state in $\mathcal{F}$ with a nominal from S. Hence, as $(\mathcal{F}, V'), w \nVdash \rho$, we have that $(\mathcal{F}, V), w \nVdash \rho[j_1/i_1, \ldots, j_n/i_n]$. But $\texttt{Pure}$ is closed under uniform substitution of nominals in S for nominals, hence $\rho[j_1/i_1, \ldots, j_n/i_n]$ belongs to $\texttt{Pure}$ and is globally satisfied in $(\mathcal{F}, V)$. We conclude that $\mathcal{F} \Vdash \texttt{Pure}$.

If $\texttt{Axiom}$ is a finite or countably infinite set of pure formulas, let $\texttt{H} + \texttt{Axiom}$ be the tableau system that results by using the formulas in $\texttt{Axiom}$ as axioms. That is, for any formula ρ in $\texttt{Axiom}$, and any nominals j, $j_1, \ldots, j_n$ that occur on a branch of a tableau, we are free to add either $j : \rho$ or $j : \rho[j_1/i_1, \ldots, j_n/i_n]$ to the end of that branch (here $i_1 \ldots, i_n$ are nominals in ρ).

Theorem 13 (Extended Strong Completeness) Let $\texttt{Axiom}$ be a finite or countably infinite set of pure formulas and let Axiom be the class of frames that $\texttt{Axiom}$ defines. Any $\texttt{H} + \texttt{Axiom}$-consistent set of formulas in a countable language is satisfiable in a countable standard model based on a frame in Axiom.

Proof I will sketch the main steps. Much as in the proof of Theorem 11, given an $\texttt{H} + \texttt{Axiom}$-consistent set of formulas Σ we will use systematic construction to build a model $\mathcal{B}$ that satisfies all formulas in Σ at a single point. But we also need to guarantee that the frame underlying $\mathcal{B}$ validates Axiom, and this means we must modify our notion of systematic tableau construction as follows.

Enumerate the axioms in $\texttt{Axiom}$ as $\rho_1, \rho_2, \rho_3, \ldots$, and let i be the new nominal we are using to start the tableau construction. Carry out stage 1 of the systematic construction as described before, and then add $i : \rho_1$ to the end of the tableau. (Thus at the end of stage 1 we no longer have a tree consisting of a single node, but a tree consisting of two nodes; the root is labelled by a formula of the form $i : \sigma$, and the second node is labelled by $i : \rho_1$.) We carry out stage $n + 1$ of the systematic construction as described before, and then, for every branch B in the tableau and every axiom $\rho \in \{\rho_1, \ldots \rho_n\}$, we add all formulas of the form $j : \rho[j_1/i_1, \ldots, j_n/i_n]$ to the end of B; here $i_1 \ldots, i_n$ are nominals that occur in ρ and $j, j_1, \ldots, j_n$ are nominals that occur on B. We then add $i : \rho_{n+1}$ to the end of every branch. In short, our new notion of systematicity is to add axioms one by one and substitute previously used nominals. At every finite stage n, every branch contains only finitely many nominals, thus this process yields another finite tree.

Suppose that B is an open branch of the tableau T obtained at the limit of this modified construction. Let $\textsc{b}$ be the set of formulas on B. As before $\textsc{b}$ is a Hintikka set, hence by Theorem 9 every formula in $\textsc{b}$ is satisfied by label in $\mathcal{B}$, the model induced by $\textsc{b}$. But $\textsc{b}$ (and hence $\mathcal{B}$) has other pleasant properties; as I shall now show, we can apply Lemma 12.

Let S_B be the set of nominals that occur in some formula in B, and let $\textsc{b}(\texttt{Axiom})$ be the smallest set containing $\texttt{Axiom}$ that is closed under substitution of nominals in S_B for nominals. Because of the way we defined the new notion of systematicity, if $\rho \in \mathrm{B}(\mathtt{Axiom})$, then for every nominal $j \in S_B$, $j : \rho$ is on branch B. But as every state in $\mathcal{B}$ is labelled by some nominal (recall that the states in $\mathcal{B}$ are simply equivalence classes of nominals that occur on B) it follows that all such formulas ρ are *globally* satisfied in $\mathcal{B}$. But this means that S_B and $\mathrm{B}(\mathtt{Axiom})$ fulfill the requirements of Lemma 12, hence $\mathrm{B}(\mathtt{Axiom})$, and hence $\mathtt{Axiom}$, is valid on the frame underlying $\mathcal{B}$.

But now the proof follows much the same lines as that of Theorem 11. Given an $\mathtt{H} + \mathtt{Axiom}$-consistent set of formulas Σ, we carry out our new systematic tableau construction. Crucially, we can show that the resulting tableau must contain at least one open branch. (As before, the key point to observe is that this can be shown by contradiction: we mimic the sequence of rule applications used to systematically build some finite stage closed tableau T_n. This part of the argument requires more care than the previous version, for we have to take the axioms into account.) Hence there must be at least one open branch B, and as the model induced by this branch validates $\mathtt{Axiom}$ and satisfies all formulas in Σ at $|i|$ we are through.

While the completeness result just stated is general, it only applies to first-order definable frame classes, for these are the only frame classes that pure formulas can define. For some purposes (for example, typical applications of feature logic) this is enough.[12] Nonetheless, it is obviously important to extend internalized labelled deduction methods to intrinsically second-order systems. I leave this for another occasion.

Tense Logic

I now show how to apply internalized labelled deduction to other modal languages by discussing the case of tense logic. I assume that the reader is familiar with tense logic and its Kripke semantics, merely commenting that I shall use the standard notation (due to Arthur Prior) in which $\Diamond$ is written as F, $\Box$ is written as G, and the symbols P and H are the backward looking counterparts of F and G respectively. We internalize the satisfaction relation just as we did for the basic modal language: by adding nominals and satisfaction operators.

So how can we define an unsigned tableau system for this language? Very straightforwardly. The rules for the booleans, the satisfaction operators, and Ref, Sym, and Nom are unchanged. The rules for F and G are simply the rules for $\Diamond$ and $\Box$ written

[12] One referee raised the following question: given that the result covers only first-order definable frame classes, why not work directly in first-order logic? Actually, at least in the short term, for some applications this may be a sensible option to explore: after all there is a wealth of expertise in first-order theorem proving. Nonetheless, the fact remains that for many applications we are working in highly-restricted, often decidable, *fragments* of first-order logic. The basic hybrid language gives us a new way of thinking about such fragments, and (in my view) in the long run the more fine-grained perspective it offers is more likely to lead to practical inference methods for many applications. I certainly believe it is worthwhile experimenting with implementations of hybrid logics for practical applications.

in Priorean notation:

$$\frac{s:F\varphi}{\begin{array}{l}s:Fa\\ a:\varphi\end{array}}\ [\mathsf{F}] \qquad\qquad \frac{\neg s:F\varphi \quad s:Ft}{\neg t:\varphi}\ [\neg\mathsf{F}]$$

$$\frac{s:G\varphi \quad s:Ft}{t:\varphi}\ [\mathsf{G}] \qquad\qquad \frac{\neg s:G\varphi}{\begin{array}{l}s:Fa\\ \neg a:\varphi\end{array}}\ [\neg\mathsf{G}]$$

For P and H, we use the mirror images of these rules (that is, these rules with F replaced by P, and G replaced by H):

$$\frac{s:P\varphi}{\begin{array}{l}s:Pa\\ a:\varphi\end{array}}\ [\mathsf{P}] \qquad\qquad \frac{\neg s:P\varphi \quad s:Pt}{\neg t:\varphi}\ [\neg\mathsf{P}]$$

$$\frac{s:H\varphi \quad s:Pt}{t:\varphi}\ [\mathsf{H}] \qquad\qquad \frac{\neg s:H\varphi}{\begin{array}{l}s:Pa\\ \neg a:\varphi\end{array}}\ [\neg\mathsf{H}]$$

Next we need to pin down the required interaction between the forward and backward looking modalities. (That is, we need to insist that these modalities are *not* independent: they exploit the same relation, but in opposite directions.) We do this by adding the following pair of rewrite rules, the *transposition* rules:

$$\frac{s:Pt}{t:Fs}\ [\mathsf{Transpose\text{-}P}] \qquad\qquad \frac{s:Ft}{t:Ps}\ [\mathsf{Transpose\text{-}F}]$$

Only one task remains: reformulating the original Bridge rule in Priorean notation and adding its backward looking counterpart:

$$\frac{s:Pt \quad t:u}{s:Pu}\ [\mathsf{Bridge\text{-}P}] \qquad\qquad \frac{s:Ft \quad t:u}{s:Fu}\ [\mathsf{Bridge\text{-}F}]$$

That's it. As an example, lets prove $p \to HFp$, a standard axiom of tense logic:

1	$\neg i:(p \to HFp)$	
2	$i:p$	$1, \neg\!\to$
$2'$	$\neg i:HFp$	Ditto
3	$i:Pj$	$2', \neg H, j$
$3'$	$\neg j:Fp$	Ditto
4	$j:Fi$	3, Transpose-P
5	$\neg i:p$	$3', 4, \neg F$
	✠ 2, 5 ✠	

The notions of systematic tableau construction and Hintikka sets can be easily adapted to prove that this system is strongly complete with respect to the class of all frames. (All that really requires checking is that Transpose-F and Transpose-P enforce the required interaction between the modalities, and they do.) Moreover, as in the modal case, this result is a stepping stone to a wide range of complete axiomatizations for other frame classes: as before, if we add *pure* formulas as axioms, completeness results are automatic.[13]

Internalized labelled deduction systems for a wide range of multi-modal logics can be obtained in much the same way as for Priorean tense logic: we internalize the Kripke satisfaction using nominals and satisfaction operators, add the required modality rules, and then try to pin down the required interaction principles.[14] But I won't explore these topics further here; instead I want to show that even stronger logics of labels can be internalized.

Binding labels

Why not *bind* the labels in our object language? To illustrate what is involved, I'll add the $\downarrow$ binder to the basic hybrid language. This enables us to bind labels to the current state: so to speak, it creates a name for the here-and-now (wherever that might happen to be).

Here are the required syntactic changes. We choose a denumerably infinite set SVAR $= \{x, y, z \ldots\}$, the set of *state variables*, disjoint from both PROP and NOM. We then stipulate that:

$$\text{WFF} := x \mid i \mid p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \rightarrow \psi \mid \Diamond\varphi \mid \Box\varphi \mid i{:}\varphi \mid x{:}\varphi \mid {\downarrow}x\varphi.$$

That is, only three changes have been made to the previous syntax: all state variables are formulas; state variables can be used to build satisfaction statements; and (crucially) we can *bind* state variables by building wffs of the form ${\downarrow}x\varphi$. In such a formula, all free occurrences of x in φ are bound by the occurrence of ${\downarrow}x$. If a formula contains no free occurrences of any variable it is called a *sentence*.[15]

Note that state variables are formulas, just as nominals are. And in the semantics presented below, state variables are going to be formulas true at precisely one state, just as nominals are. So why bother drawing a distinction between state variables and nominals? Why not directly bind nominals with $\downarrow$? Well, we *could* have done things this way — but I find it neater to have two types of label: those which can be bound (state variables) and those which cannot (nominals). In effect, nominals will a play a role for us analogous to that of *parameters* in first-order proof theory.

Now for the semantics. We need a way to cope with the interpretation of bound variables, and a convenient way to do this is to make use of Tarski-style assignment

[13] And we can do some interesting new things in the tense logical case. For example, the class of Church-Rosser frames is not definable in the basic hybrid language, but it *is* definable in the hybrid tense language, and we don't even have to use satisfaction operators:

$$Fi \wedge Fj \rightarrow F(i \wedge FPj).$$

But we pay a price for this expressivity. The satisfaction problem for ordinary Priorean tense logic over arbitrary frames is PSPACE-complete, but Maarten Marx has given an extremely simple and elegant proof which shows that adding even one nominal to tense logic yields a system with an EXPTIME-hard satisfaction problem over arbitrary frames; you can find the proof in [2].

[14] Above we added extra rewrite rules, namely Transpose-F and Transpose-P, to capture the required tense logical interactions, but we could instead have made use of the (pure) axioms $i \rightarrow HFi$ and $i \rightarrow GPi$. More generally, when working with multimodal languages in which the required interaction principle can be captured using pure axioms, completeness is immediate by Lemma 12 (the number of modalities the underlying modal language contains is irrelevant to the proof of this lemma).

[15] Precise definitions of such concepts as *free*, *bound*, *substitution*, and so on can be found in [9]. But they're not needed here; experience with classical logic or lambda calculus is a reliable guide.

functions. So, given a model $\mathcal{M} = (W, R, V)$, an *assignment* on $\mathcal{M}$ is a function $g : \mathrm{SVAR} \longrightarrow W$. (Thus an assignment makes a state variable true at precisely one state.) We then relativize the clauses of the satisfaction definition for the basic hybrid language to assignments, and add the three new clauses we require. Here's a selection:

$$
\begin{array}{lll}
\mathcal{M}, g, w \Vdash x & \text{iff} & w = g(x) \text{ where } x \in \mathrm{SVAR} \\
\mathcal{M}, g, w \Vdash x{:}\varphi & \text{iff} & \mathcal{M}, g, g(x) \Vdash \varphi \\
\mathcal{M}, g, w \Vdash \varphi \wedge \psi & \text{iff} & \mathcal{M}, g, w \Vdash \varphi \text{ and } \mathcal{M}, g, w \Vdash \psi \\
\mathcal{M}, g, w \Vdash \Diamond\varphi & \text{iff} & \exists w'(wRw' \ \& \ \mathcal{M}, g, w' \Vdash \varphi) \\
\mathcal{M}, g, w \Vdash {\downarrow}x\varphi & \text{iff} & \mathcal{M}, g', w \Vdash \varphi, \text{ where } g' \overset{x}{\sim} g \text{ and } g'(x) = w
\end{array}
$$

The first four clauses are obvious generalizations of those we have seen before. The fifth does something new: it defines $\downarrow$ to be an operator that binds variables to the state w at which evaluation is being performed. The notation $g' \overset{x}{\sim} g$ means that g' is the assignment that differs from g, if at all, only in what it assigns to x. By stipulating that $g'(x)$ is to be w, we bind a label to the here-and-now. A formula of this language is valid if it is globally satisfied in any standard model under any assignment.

The resulting language is attractive. For example, $\downarrow$ cooperates smoothly with the satisfaction operators to enable us to define *Until*:

$$\mathit{Until}(\varphi, \psi) \ := \ {\downarrow}x(\Diamond{\downarrow}y(\varphi \wedge x{:}\Box(\Diamond y \rightarrow \psi))).$$

But as this language has been discussed at length in [7], [11], [12] and [2] (and see [27] for a stronger system containing $\downarrow$), let's turn straight to the question of providing it with an internalized labelled deduction system.[16]

The pleasant thing is, there's not much to say. First, we can re-use our original tableau rules for the booleans, satisfaction operators, and modalities, and also our rewrite rules, by making one small change: we should now interpret the metavariables s, t and u in these rules as ranging over both nominals *and* state variables. Note, however, that in the statement of the existential rules, a will continue to be interpreted as a new *nominal*. Doing so means that we don't have to worry about a getting accidentally bound (that is, we really are using nominals as parameters).

Second, we add the $\downarrow$-rule and the $\neg\downarrow$-rule:

$$\frac{s{:}{\downarrow}x\varphi}{s{:}\varphi[s/x]}\ [\downarrow] \qquad\qquad \frac{\neg s{:}{\downarrow}x\varphi}{\neg s{:}\varphi[s/x]}\ [\neg\downarrow]$$

(Here $\varphi[s/x]$ means substitute s for all free occurrences of x in φ. If s is a variable, before substituting we rename bound occurrences of s in φ to prevent accidental capture.)

[16] A number of results about this language have been proved recently. For example, Maarten Marx and I, using different techniques, have shown that sentences of the $\downarrow$-enriched basic hybrid language pick out precisely the fragment of the first-order correspondence language that is invariant under generated submodels, and Carlos Areces (using a model-theoretic argument) has proved that interpolation holds for this language, and hence that the Beth property holds too. These results (and others) are gathered together in [2]. Jerry Seligman has recently reported a constructive (sequent calculus based) proof of interpolation. It is now clear that the $\downarrow$ operator occupies a fundamental position in the space of hybrid languages.

Here's an example. Like the satisfaction operators, $\downarrow$ is self dual. That is,

$$\downarrow x\varphi \leftrightarrow \neg\downarrow x\neg\varphi$$

is valid. The left to right direction of this equivalence can be proved as follows (the other direction is similar):

1	$\neg i : (\downarrow x\varphi \to \neg\downarrow x\neg\varphi)$	
2	$i : \downarrow x\varphi$	$1, \neg\to$
$2'$	$\neg i : \neg\downarrow x\neg\varphi$	Ditto
3	$i : \downarrow x\neg\varphi$	$2', \neg\neg$
4	$i : \neg\varphi[i/x]$	$3, \downarrow$
5	$\neg i : \varphi[i/x]$	$4, \neg$
6	$i : \varphi[i/x]$	$2, \downarrow$
	✠ 5, 6 ✠	

Here's another example: $\downarrow xx$ is valid, and we can prove it as follows:

1	$\neg i : \downarrow xx$	
2	$\neg i : i$	$1, \neg\downarrow$
3	$i : i$	Ref
	✠ 2, 3 ✠	

Strong completeness with respect to the class of all frames can be shown by modifying the notions of systematic tableau construction and Hintikka sets appropriately. Moreover Theorem 13 can be extended to cover any pure axiomatic extension of this basic system. As before, "pure" simply means "contains no propositional variables", so pure formulas may contain state variables and the $\downarrow$ binder, and this gives us a wide range of strong completeness results.[17] The system can be further generalized. For example, several authors have discussed very strong (essentially first-order) systems in which it is possible to quantify classically across state variables using $\exists$ and $\forall$ (for example [13], [25], [27], [9], and [10]). Internalized deduction can be defined for such languages too — indeed, this is what Jerry Seligman did (for languages of arbitrary signature) in his pioneering work on internalized labelled deduction.

8 Related work

I will now discuss related work, namely Seligman-style sequent systems, and Tzakova-style tableau systems. These have much in common with the methods described here, but there are some interesting differences, and one is particularly important: Tzakova, and Seligman in his most recent work, both employ what I call *mixed methods*. That is, they use labelling in both the object language and the metalanguage.[18]

[17] For more details, see [11, 12]. These papers deal with Hilbert systems and use maximal consistent sets of formulas rather than Hintikka sets, but the underlying arguments are much the same.

[18] At the time of writing the final versions of [43] and [48] weren't available, and what follows is largely based on discussions with these authors. It is provisional, and doesn't cover all aspects of their work.

Seligman-style sequent systems

In a series of early papers (see [41, 42, 43]) Jerry Seligman investigated the proof theory of hybrid languages containing satisfaction operators in which it is possible to classically quantify over nominals.[19] The early papers explore internalized label deduction for these systems. In his most recent work (see [44]) Seligman has extended his work to cover the languages discussed in this paper (and indeed, some others) and shifted from internalized deduction methods to mixed methods in which the labelling is used in the metalanguage as well.

The best approach to Seligman's work is to return to the sequent-based presentation of internalized labelled deduction given at the end of Section 3. All Seligman's systems are related to this, but there are some differences. The most obvious concerns the way he captures the modal theories of state equality and state succession. Seligman does not use a rewrite system to do this: rather he uses *substitution*. Here are simplified forms of two of Seligman's key rules:

$$\frac{[s/t]\Gamma \longrightarrow [s/t]\Sigma}{s:t, \Gamma \longrightarrow \Sigma}\ \textsf{[SubL]} \qquad\qquad \frac{[t/s]\Gamma \longrightarrow [t/s]\Sigma}{s:t, \Gamma \longrightarrow \Sigma}\ \textsf{[SubR]}$$

That is, satisfaction statements of the form $s:t$ license the uniform substitution of s by t (and t by s) throughout a sequent. As an example of these rules in action, here's how to show that

$$\Diamond(i \wedge \varphi), \Diamond(i \wedge \psi) \longrightarrow \Diamond(\varphi \wedge \psi)$$

is a valid sequent; it's probably best to read the proof from bottom to top. The symbols j, k, and l are taken to be new nominals.

$$
\begin{array}{cl}
\overline{j:\Diamond i, i:\varphi, i:\psi, \longrightarrow i:\psi} \qquad \overline{j:\Diamond i, i:\varphi, i:\psi, \longrightarrow i:\varphi} & \wedge\mathrm{R} \\
\hline
j:\Diamond i, i:\varphi, i:\psi, \longrightarrow i:(\varphi \wedge \psi) & \Diamond\mathrm{R}(i) \\
\hline
j:\Diamond i, i:\varphi, i:\psi, \longrightarrow j:\Diamond(\varphi \wedge \psi) & \mathrm{WL} \\
\hline
j:\Diamond i, i:i, i:\varphi, j:\Diamond i, i:i, i:\psi, \longrightarrow j:\Diamond(\varphi \wedge \psi) & \mathrm{SubL}[i/l] \\
\hline
j:\Diamond i, i:i, i:\varphi, j:\Diamond l, l:i, l:\psi, \longrightarrow j:\Diamond(\varphi \wedge \psi) & \mathrm{SubL}[i/k] \\
\hline
j:\Diamond k, k:i, k:\varphi, j:\Diamond l, l:i, l:\psi, \longrightarrow j:\Diamond(\varphi \wedge \psi) & \wedge\mathrm{L}(\times 2) \\
\hline
j:\Diamond k, k:(i \wedge \varphi), j:\Diamond l, l:(i \wedge \psi), \longrightarrow j:\Diamond(\varphi \wedge \psi) & \Diamond\mathrm{L}(l) \\
\hline
j:\Diamond k, k:(i \wedge \varphi), j:\Diamond(i \wedge \psi) \longrightarrow j:\Diamond(\varphi \wedge \psi) & \Diamond\mathrm{L}(k) \\
\hline
j:\Diamond(i \wedge \varphi), j:\Diamond(i \wedge \psi) \longrightarrow j:\Diamond(\varphi \wedge \psi) &
\end{array}
$$

(Note that each of the sequents at the leaf nodes contains an identical formula to the right and the left of the sequent arrow, hence repeated application of weakening reduces these sequents to axioms.)

[19] The 1997 publication date of [43] is misleading: the paper was written much earlier and is essentially the full version of the 1991 and 1992 technical reports. It gives a good overview of Seligman's technical work on sequent calculus and natural deduction for strong (first-order equivalent) hybrid language. In addition, it convincingly motivates such languages from the point of view of an Austinian theory of meaning; in effect Seligman argues that the Austinian theory demands strongly contextualized languages, and that internalizing labels contextualizes modal languages in the required way. But while the 1997 paper is the key article, try to get hold of the 1991 report as well; it contains interesting material that didn't make it into the final version.

The reader may find it interesting to redo this proof using the rewrite rules presented earlier. If you do, I think you will agree that the Seligman-style substitution-based proof is superior; substituting throughout a sequent is natural, whereas stepwise applying all the needed rewrites to the sequents is tedious. On the other hand, if you formulate Seligman-style substitution rules for the tableau system, I think you will find substitutions less convenient than rewrites, at least for hand calculation. Stepwise extension of branches is the name of the game in handwritten tableau proofs; global updates, such as substitutions are less pleasant to work with.

Seligman gives a detailed proof-theoretical analysis of substitution in hybrid languages (the reader who worked through the proof of Theorem 13 will realize why this is useful). This analysis is a Gentzen-style cut elimination proof. However because hybrid languages embody *theories* (namely, theories of state equality and state succession), cut elimination is far from trivial; indeed, Seligman proceeds by showing the proof-theoretical equivalence of a range of different sequent systems. One spinoff of his analysis is a number of general completeness proofs, similar in range and spirit to the ones discussed in this paper. As he shows, it is simple to prove completeness results for his sequent systems by using the cut rule. Then, having established completeness for such systems, he appeals to his cut-elimination results to transfer the result to the cut-free systems.

But his most recent work introduces a more fundamental difference: in the sequent rules, the symbol $s:\varphi$ is *not* a pure object language expression: rather, it is *metalinguistic* labelling symbol s: applied to object language formula φ. The object language formula φ may well contain nominals and perhaps occurrences of object-level satisfaction operators as well. That is, Seligman's later work revolves around sequent calculi containing metalinguistic labelling devices for working with object languages containing nominals. This mixed approach to labelling is interesting, but I want to approach it via the work of Miroslava Tzakova.

Tzakova-style tableaux

Miroslava Tzakova has explored tableau systems for a variety of hybrid languages. Her approach differs from the unsigned systems presented here (and from Seligman-style labelled sequent calculi) in a number of ways, however I will confine my discussion to just one aspect of her work: the use of labels in both the object language and the metalanguage.

Let's see how to obtain a Tzakova-style tableau system from the systems of this paper. Here are the $\Diamond$ rules given above:

$$\frac{s:\Diamond\varphi}{\begin{array}{c}s:\Diamond a\\ a:\varphi\end{array}} \qquad\qquad \frac{\neg s:\Diamond\varphi \quad s:\Diamond t}{\neg t:\varphi}$$

As we have already observed, these rules can be turned into signed rules by enriching the metalanguage with the symbols $\mathcal{T}$ for true and $\mathcal{F}$ for false:

$$\frac{\mathcal{T}s:\Diamond\varphi}{\begin{array}{c}\mathcal{T}s:\Diamond a\\ \mathcal{T}a:\varphi\end{array}} \qquad\qquad \frac{\mathcal{F}s:\Diamond\varphi \quad \mathcal{T}s:\Diamond t}{\mathcal{F}t:\varphi}$$

To get to a Tzakova-style tableau system, we take this line of thought a stage further: we add labels to the metalanguage and thereby generalize the underlying proof theoretic ideas to languages that don't contain satisfaction operators. Let's prefix each formula in a tableau proof by a pair whose first element is a sign, and whose second element is a label (we'll assume that object language and metalanguage have access to the same set of labels; in fact, we'll simply use our supply of nominals for metalinguistic purposes as well). When a formula φ is prefixed by a pair $(\mathcal{T}, s)$, this will mean that φ is true at the point labelled s. On the other hand, $(\mathcal{F}, s)\ \varphi$ will mean that φ is false at s. This yields the following $\Diamond$ rules:

$$\frac{(\mathcal{T}, s)\Diamond\varphi}{\begin{array}{c}(\mathcal{T}, s)\Diamond a\\ (\mathcal{T}, a)\varphi\end{array}} \qquad\qquad \frac{(\mathcal{F}, s)\Diamond\varphi \quad (\mathcal{T}, s)\Diamond t}{(\mathcal{F}, t)\varphi}$$

Clearly these rules trade on the same ideas that the previous version do — but they do so without assuming that the object language contains satisfaction operators. In effect, the key ideas have been lifted into the metalanguage. And this can be useful. To give one example, Tzakova has defined a mixed tableau system which handles the $\downarrow$ binder smoothly in the absence of satisfaction operators.

But perhaps the most important thing to stress about Tzakova's strategy (and Seligman's later strategy too, for that matter) is that it is *not* just classic Gabbay-style labelled deduction: it makes crucial use of the nominals in the object language. That is, just as in the systems discussed in this paper, the core discipline that drives the (very different) systems of Tzakova and Seligman stems from the fact that they treat labels as formulas in the object language, and thus endow labels with a logic. That is, both approaches treat labels as first class citizens, even though they don't insist on having satisfaction operators in the language. The presence of both object-level and meta-level labels makes easy communication possible between the levels, and frees Tzakova and Seligman from the need to define a full Gabbay-style metalinguistic labelling discipline: the meta-labels can exploit the logic of labels present in the object language. Mixed methods are an interesting tradeoff between the extremes of Gabbay-style labelled deduction and internalized labelled deduction.

Summing up, there are actually *three* approaches to labelled deduction:

External The classic Gabbay-style approach. Labels are only provided in the metalanguage, which is boosted with some sort of labelling discipline (typically an algebra of labels which controls the proof process).

Mixed Typified by Seligman [43] and Tzakova [48]. Labels belong in *both* the metalanguage and the object language (where they are treated as formulas). The presence of nominals in the object language provides the core discipline, and frees us from having to provide a full blown metalinguistic labelling algebra, while the presence of a metalinguistic labelling mechanism enables us to make fewer assumptions about the object language. The two levels talk to each other via the shared labels.

Internalized As in this paper and the earlier work of Seligman. Labels are only provided in the object language, which is further boosted by the inclusion of satisfaction operators which internalize the labelling process. Permits use of standard proof methods, with no new metalinguistic machinery, for the required labelling discipline has been fully internalized into the object language.

In my view the existence of this deductive spectrum is yet another line of evidence pointing to the following conclusion: *labelling is not merely an interesting option in modal logic, it is fundamental.* To close this paper I want to defend this claim.

9 Learning to love labels

The idea of naming states in modal logic and related formalisms has been in the air for some time. As well as the hybrid logic and labelled deductive system traditions, there is the use of re-entrancy marking tags in AVMs and other feature description formalisms (see, for example, [45]), the use of the A-box and the $\mathcal{O}$ operator ("one of") in description logics (see, for example, [19]) and the Polish tradition of modal logics for rough sets (see, for example, [33]). Moreover, one recent approach to general modal proof theories (see [3]) combines modal languages with (parts of) their correspondence languages, thereby allowing the object language access to labels.

But in spite of this convergence, many modal logicians seem reluctant to embrace labelling. Labelling, it is conceded, may be useful for some purposes, but it is essentially alien to modal logic, and the theoretician can afford to ignore it. In my view, this is profoundly mistaken. Far from being an intrusion, labelling, and indeed, a generalization of labelling called *sorting*, is woven into the very fabric of modal logic.

At first glance, propositional modal logic seems to offer us a rather limited range of options. Its syntax is extremely simple, as are the Kripke models on which it is interpreted. Certainly most extended (propositional) modal logics are based on a single idea: expanding the syntax by adding more modal operators. In fact, the idea of adding new modal operators is now so deeply ingrained that it has come to seem the *only* legitimate form of extension.

But it's not — and that was Arthur Prior's insight. Propositional modal logics are all about manipulating *propositions*, abstract bearers of information. But there are many different kinds of information, and some kinds (for example, propositions true at a unique state) bring important logical ideas (such as identity) into play. And these ideas *can* be pinned down: Prior did so by *sorting* the propositional variables, so that different types of information were *explicitly* marked in the object language. This simple mechanism makes it possible to explore the logical impact of different types of information.

Prior's sorting strategy underlies the work of this paper. We introduced a second sort of atomic symbol (nominals), insisted they be true at one state in every model, and introduced a collection of new normal modal operators (the satisfaction operators) to exploit them to the full. The changes involved were simple, nonetheless we learned that our hybrid language could do useful work for us: it let us define new classes of frames and construct intuitive proof-theories. Actually, we could have been bolder and added further sorts. For example, as long ago as 1970, Robert Bull, inspired by Arthur Prior's work, introduced a *three*-sorted language: it contained ordinary propositional variables (the "arbitrary information" sort) nominals (the "labelling sort") and atomic symbols true at precisely the points on some path through the frame (the "course of history" sort).[20] But let's postpone such adventures for another occasion: the important point to note is that nothing we did in this paper (and nothing Prior and Bull did in their work either), is alien to modal logic. The basic hybrid language manipulates propositions with the aid of operators, and that is what modal logic is all about.

The history of modal logic is essentially the story of relatively rigid forms of semantic analysis (for example, the use of state descriptions in Carnap [15]) giving way to more flexible accounts (notably the relativized semantics of Kripke [31, 32], and a decade later the general frames of Thomason [47]). In fact most of the semantical freedom we currently enjoy stems either from the *Kripkean parameter* (the idea that by making the transition relation R explicit and varying its properties we can control logics) or the *Thomasonian parameter* (defining validity not in terms of *all* valuations on a frame, but in terms of some well-behaved subcollection).

Prior introduced a third semantic parameter when he invented hybrid languages. Like the Thomasonian parameter, the *Priorean parameter* is based on the idea of restricting the available valuations, but it does so differently, and with different aims in mind: the point is to make explicit the different sorts of proposition we are manipulating, and to identify their logic. Now, this idea gives rise to a vision which seems worthy of further exploration: modal logic as a genuine theory of information, with the Kripkean parameter controlling how information is distributed, while the Priorean parameter pins down the logic of different information types. But it also has something more down-to-earth to tell us, namely the central message of the present paper: labelling, and labelled deduction, are part and parcel of the modal enterprise.

Acknowledgments

I owe an immense debt to Jerry Seligman for a decade of discussion of hybrid languages, and for insisting on the relevance of his proof-theoretic analyses; he was right all along, and it took me a long time to see it. I am also extremely grateful to Miroslava Tzakova; our collaboration on Hilbert systems for hybrid languages was the catalyst for this paper. Carlos Areces provided me with detailed comments on two earlier drafts of this paper, pointing out many typos, infelicities, and errors; thanks to his efforts, the final version is a lot more readable than the first. I am also grateful to Stéphane Demri and Rajeev Goré for their comments on the penultimate version. Finally, I want to thank Manfred Pinkal for funding a visit by Jerry Seligman to the University of Saarland, and the two referees for their comments.

References

[1] J. Allen (1984). Towards a general theory of knowledge and action. *Artificial Intelligence*, **23**, 123–154.

[2] C. Areces, P. Blackburn, and M. Marx (1999). Hybrid logics. Characterization, Interpolation and Complexity. CLAUS-Report 104, Computerlinguistik, University of Saarland, `http://www.coli.uni-sb.de/cl/claus`.

[20] See [13]. For further discussion of the "course of history" sort, see [28], [11], [12]. For the use of sorting to deal with intervals, see [5].

[3] D. Basin, S. Matthews, L. Viganò (1997). Labelled propositional modal logics: Theory and practice. *Journal of Logic and Computation*, **7**, 685–717.

[4] J. van Benthem (1993). Programming operations that are safe for bisimulations. CSLI Report 93-179, Center for the Study of Language and Information, Stanford University.

[5] P. Blackburn (1990). *Nominal Tense Logic and Other Sorted Intensional Frameworks*. PhD Thesis, Centre for Cognitive Science, University of Edinburgh.

[6] P. Blackburn (1993). Nominal tense logic. *Notre Dame Journal of Formal Logic*, **14**, 56–83.

[7] P. Blackburn and J. Seligman (1995). Hybrid languages. *Journal of Logic, Language and Information*, **4**, 251–272.

[8] P. Blackburn and J. Seligman (1998). What are hybrid languages? In M. Kracht, M. de Rijke, H. Wansing, and M. Zakharyaschev (eds.), *Advances in Modal Logic, Volume 1*, pages 41–62. CSLI Publications, Stanford University.

[9] P. Blackburn and M. Tzakova (1998). Hybrid completeness. *Logic Journal of the IGPL*, **4**, 625–650. `http://www.oup.co.uk/igpl`.

[10] P. Blackburn and M. Tzakova (1998). Hybridizing concept languages. *Annals of Mathematics and Artificial Intelligence*, **24**, 23–49.

[11] P. Blackburn and M. Tzakova (1999). Hybrid languages and temporal logic. *Logic Journal of the IGPL*, **7**, 27–54. `http://www.oup.co.uk/igpl`.

[12] P. Blackburn and M. Tzakova (1998). Hybrid languages and temporal logic (full version). CLAUS-Report 96, Computerlinguistik, University of Saarland, `http://www.coli.uni-sb.de/cl/claus`.

[13] R. Bull (1998). An approach to tense logic. *Theoria*, **36**, 282–300.

[14] S. Buvač, V. Buvač and I. Mason (1998). Metamathematics of contexts. *Fundamenta Informaticae*, **23**, 263–301.

[15] R. Carnap (1946). Modalities and quantification. *Journal of Symbolic Logic*, **11**, 33–64.

[16] M. Cresswell (1990). *Entities and Indices.* Kluwer.

[17] S. Demri (1999). Sequent calculi for nominal tense logics: a step towards mechanization? To appear in N. Murray (ed.), *Conference on Tableaux Calculi and Related Methods (TABLEAUX), Saratoga Springs, USA*, Springer Verlag, LNAI 1617.

[18] S. Demri and R. Goré (1999). Cut-free display calculi for nominal tense logics. To appear in N. Murray (ed.), *Conference on Tableaux Calculi and Related Methods (TABLEAUX), Saratoga Springs, USA*, Springer Verlag, LNAI 1617.

[19] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf (1996). Reasoning in description logics. In G. Brewka (ed), *Principles of Knowledge Representation*, Studies in Logic, Language and Information, CSLI Publications, Stanford University.

[20] M. Fitting (1983). *Proof Methods for Modal and Intuitionistic Logics*. Reidel.

[21] D. Gabbay (1996). *Labelled Deductive Systems.* Oxford University Press.

[22] D. Gabbay (1981). An irreflexivity lemma. In U. Mönnich (ed.), *Aspects of Philosophical Logic*, pages 67–89. Reidel.

[23] G. Gargov and V. Goranko (1993). Modal logic with names. *Journal of Philosophical Logic*, **22**, 607–636.

[24] G. Gargov and S. Passy (1988). Determinism and Looping in Combinatory PDL. *Theoretical Computer Science*, **61**, 259–277.

[25] G. Gargov, S. Passy and T.Tinchev (1987). Modal environment for boolean speculations (preliminary report). In D. Skordev (ed.), *Mathematical Logic and its Applications. Proceedings of the Summer School and Conference dedicated to the 80th Anniversary of Kurt Gödel, Druzhba, 1986*, pages 253–263. Plenum Press.

[26] V. Goranko (1994). Temporal logic with reference pointers. In D. Gabbay and H. Ohlbach (eds.), *Proceedings of the 1st International Conference on Temporal Logic*, volume 827 of *LNAI*, pages 133–148. Springer.

[27] V. Goranko (1996). Hierarchies of modal and temporal logics with reference pointers. *Journal of Logic, Language and Information*, **5**, 1–24.

[28] V. Goranko (1996). An interpretation of computational tree logics into temporal logics with reference pointers. Technical Report 2/96, Verslagreeks van die Department Wiskunde, RAU, Department of Mathematics, Rand Afrikaans University, Johannesburg, South Africa.

[29] P. Hasle (1991). Building a temporal logic for natural language understanding with the HOL-system. In C. Brown and G. Koch (eds.), *Natural Language Understanding and Logic Programming III*, Elsevier Science Publishers.

[30] W. Hodges (1977). *Logic*. Penguin.

[31] S. Kripke (1959). A completeness theorem in modal logic. *Journal of Symbolic Logic*, **24**, 1–14.

[32] S. Kripke (1963). Semantical considerations on modal logic. *Acta Philosphica Fennica*, **16**, 83–94.

[33] B. Konikowska (1997). A logic for reasoning about relative similarity. *Studia Logica*, **58**, 185–226.

[34] R. Ladner (1977). The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, **6**, 467–480.

[35] J. McCarthy and P. Hayes (1969). Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, (eds.), *Machine Intelligence 4*, Edinburgh University Press.

[36] S. Passy and T. Tinchev (1985). Quantifiers in combinatory PDL: completeness, definability, incompleteness. In *Fundamentals of Computation Theory FCT 85*, volume 199 of *LNCS*, pages 512–519. Springer.

[37] S. Passy and T. Tinchev (1991). An essay in combinatory dynamic logic. *Information and Computation*, **93**, 263–332.

[38] A. Prior (1967). *Past, Present and Future.* Oxford University Press, Oxford.

[39] A. Prior and K. Fine (1977). *Worlds, Times, and Selves.* University of Massachusetts Press.

[40] N. Rescher and A. Urquhart (1971). *Temporal Logic*. Springer Verlag.

[41] J. Seligman (1991). A cut-free sequent calculus for elementary situated reasoning. Technical Report HCRC-RP 22, HCRC, Edinburgh.

[42] J. Seligman (1992). Situated consequence for elementary situation theory. Logic Group Preprint IULG-92-16, Indiana University.

[43] J. Seligman (1997). The logic of correct description. In M. de Rijke (ed.), *Advances in Intensional Logic*, pages 107–135. Applied Logic Series, Kluwer.

[44] J. Seligman (1998). Proof theory for contextual reasoning. Manuscript.

[45] G. Smolka (1992). Feature constraint logics for unification grammars. *Journal of Logic Programming*, **12**, 51–87.

[46] R. Smullyan (1968). *First-Order Logic.* Springer-Verlag.

[47] S. Thomason (1972). Semantic analysis of tense logic. Journal of Symbolic Logic, **37**,150–158.

[48] M. Tzakova (1999). Tableaux calculi for hybrid logics. To appear in N. Murray (ed.), *Conference on Tableaux Calculi and Related Methods (TABLEAUX), Saratoga Springs, USA*, Springer Verlag, LNAI 1617.