Modal Logic and Attribute Value Structures

Patrick Blackburn

Abstract

This paper shows that there is a close correspondence between propositional modal logic and the AV formalisms of computational linguistics. A particularly important aspect of this relationship is that unification can be seen as testing for modal satisfiability. The paper considers three modal languages — L, L^N and L^{KR} — and for each of them describes the correspondence involved and proves results concerning completeness, decidability and expressive power.

This paper examines the relationship between various languages of modal logic and an approach to the specification and processing of natural language grammars currently popular in computational linguistics. This approach is the use of Attribute Value formalisms, and the main aims of the paper are to show that the most common Attribute Value formalisms are nothing but languages of propositional modal logic, and to establish the basic logical theory of the languages concerned.

The first section is an overview of the main ideas of the Attribute Value approach to grammar. We discuss the concepts of attributes, values and unification — first in terms of Attribute Value Matrices, and then in terms of certain labeled decorated graphs — and from this discussion it emerges that Attribute Value Structures are essentially a certain type of Kripke model. Thus Attribute Value Structures *can* be described using propositional modal logic; the key assertion of this paper is that this is what computational linguists actually do.

The demonstration begins in the second section. We note that Attribute Value Matrices directly correspond to wffs of the language L, the simplest modal language for talking about Attribute Value Structures. We then sketch some of the basic logical properties of L, and outline the technique for establishing decidability that will be used throughout the paper.

In the third section we consider the *lingua franca* of Attribute Value formalisms: Attribute Value Matrices augmented by 'boxlabels'. These turn

out to amount to wffs in the modal language L^N, which is L augmented by nominals. Some results concerning L^N are noted.

In the fourth section we consider the interface language of the PATR-II system, and show how the modal language L^{KR} can be abstracted from it. In fact L^{KR} — the language of Kasper Rounds logic — has been considered before in the Attribute Value literature. Intriguingly, however, it's passed largely unremarked that it really is a modal language (and a very beautiful one at that), and its logical theory has hardly been explored. This section to some extent rectifies the omission.

We close the paper with some brief remarks concerning further work.

1 Attribute Value Structures

During the mid 1970s a number of theories of generative grammar were developed that broke with the dominant Chomskyan paradigm. At around the same time, computational linguists began to seek more principled ways of representing and processing linguistic information. These two research currents came to exert considerable mutual influence, and something like a consensus began to emerge. The consensus was this: *Attribute Value Structures are a valuable way of representing linguistic information, and the unification of such structures is a fundamental part of the parsing process.* Insofar as current computational linguistics has stable orthodoxies, something along these lines ranks as one.

So what are Attribute Value Structures, or Feature Value Structures as they are sometimes called? One of the most common ways of thinking about them is in terms of Attribute Value Matrices (AVMs). The computational linguist working with some linguistic theory is thought to have at his or her disposal a collection of atomic entities (or constants) and a collection of attributes (or features). The choice of atomic entities and attributes varies widely from theory to theory, but typical of the atomic entities a syntactician might use are *3rd, 2nd, 1st, singular, plural* and *nominative*, and the same person might employ a collection of attributes that included PERSON, NUMBER, TENSE, CASE, and AGREEMENT. Using these entities the computational linguist builds up AVMs as follows. First, any atomic entity taken alone is an AVM. Such AVMs are called atomic AVMs, and there are no other atomic AVMs than these. For example, the simple specification *1st* is an atomic AVM. Second, any attribute and any atomic AVM can be enclosed in that order between square brackets and the result is another AVM. We

say that the constituent attribute has the value (or takes the value) of the atomic entity concerned in the new AVM. For example, we can combine the attribute PERSON with the atomic AVM *1st* to form the AVM [PERSON *1st*], and in this AVM we say that the attribute PERSON takes the value *1st*. Third, any attribute and any AVM can be enclosed in that order between square brackets, and the result is another AVM. In short, attributes can take complex values. For example, [AGREEMENT [NUMBER *plural*]] is an AVM where the attribute AGREEMENT takes the complex value [NUMBER *plural*]. Finally, given a finite collection of *non-atomic* AVMs, a new AVM may be formed by enclosing the collection in square brackets, and then erasing the outermost brackets of the enclosed AVMs. For example, from the AVMs [AGREEMENT [PERSON*1st*]] and [CASE *nominative*] we can form the AVM

$$\begin{bmatrix} \text{AGREEMENT} & [\text{PERSON } \textit{1st}] \\ \text{CASE} & \textit{nominative} \end{bmatrix}$$

As this example shows, when this last rule of combination is used the convention is to list the combined AVMs vertically rather than horizontally. Note that this last mode of combination is restricted to non-atomic AVMs. For example, we *cannot* stack *3rd* on top of *plural* to form a new AVM (to do so would be to fall foul of what computational linguists call the 'constant-constant clash'), nor can we stack *plural* on top of [CASE *nom*] (here we'd be falling prey to a 'constant-compound clash'). We'll meet these restrictions again shortly in a new guise, however more immediately pressing is the fact that there is an interesting restriction on this last mode of combination that hasn't yet been stated. It is this: *if any two distinct elements in the collection are pairs whose first component is the same attribute, we cannot form a new AVM out of the collection.* For example, from [PERSON *1st*] and [PERSON *2nd*] we *cannot* form

$$\begin{bmatrix} \text{PERSON} & \textit{1st} \\ \text{PERSON} & \textit{2nd} \end{bmatrix}$$

These four rules are the only way to build AVMs, thus AVMs are a fairly simple kind of inductively defined structure. But what is the intuition underlying them? Essentially it's that of partial functional dependency between attributes and possibly complex values. It is this interpretation that motivates the interesting restriction on the fourth construction rule: the restriction is essentially a functionality requirement. Moreover it is precisely this notion of capturing partial functional dependencies between attributes

and values that makes AVMs (and their various cousins) so attractive to grammar writers: the whole business of writing grammars for natural languages can be construed in these terms, and it turns out that this construal can yield perspicuous analyses.

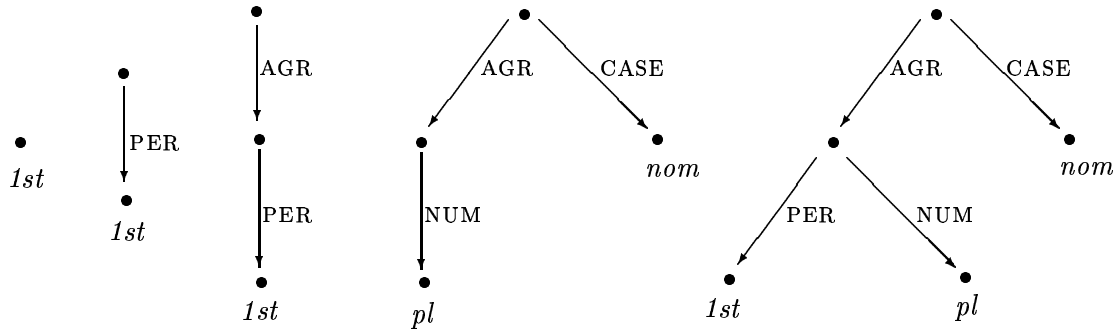
Now that we know what AVMs are, and something of the intuitions they attempt to capture, we must consider how they are used. The key concept here is *unification*. Unifying two AVMs means forming another AVM that combines all the information about partial functional dependencies embodied in the two constituents. For example, writing $\sqcup$ for the unification process, we have that:

$$[\textsc{agr}\ [\textsc{num}\ \textit{plural}]] \sqcup \begin{bmatrix} \textsc{agr} & [\textsc{per}\ \textit{1st}] \\ \textsc{case} & \textit{nominative} \end{bmatrix} = \begin{bmatrix} \textsc{agr} & \begin{bmatrix} \textsc{num} & \textit{plural} \\ \textsc{per} & \textit{1st} \end{bmatrix} \\ \textsc{case} & \textit{nominative} \end{bmatrix}$$

There is a clear sense in which the AVM on the right hand side embodies all the information in the two constituent structures; it is the result of unifying these structures. Note that it is not always possible to unify two AVMs, as the information they embody may be incompatible. In effect we have already seen a simple example of this: we cannot unify [PERSON *2nd*] and [PERSON *1st*], for there is no AVM that embodies these (conflicting) claims.

Unification is another reason why many computational linguists find Attribute Value approaches to grammar so attractive: parsing such a grammar essentially boils down to unification. This gives a very natural picture of what the business of natural language parsing is all about — it is 'the coherent merging of underspecified information' — a picture that becomes positively addictive when attempts are made to move beyond the realms of pure syntax and integrate semantic, morphological and phonological information.

Now the discussion of AVMs has revealed some of the reasons why thinking in terms of attributes, values and unification is so popular in computational linguistics, but it hasn't explicitly brought out two key ideas about Attribute Value Structures that most computational linguists share: that Attribute Value Structures are certain kinds of *decorated labeled graphs*, and that unification is the partial operation of coherently merging these structures. Let's consider some simple examples which show how AVMs give rise to such graphs, and what unification means in graphical terms.



(i) (ii) (iii) (iv) (v)

The first graph, consisting of a single node decorated with the atomic information *1st*, corresponds to the atomic AVM *1st*. More generally, atomic AVMs correspond to decorations of nodes, and in particular to decorations of terminal nodes. The second graph corresponds to the AVM [PERSON *1st*]. Note that the atomic AVM *1st* decorates a terminal node. Moreover note that the attribute PERSON labels a transition. Again this is perfectly general: attributes correspond to transition labels. Next consider the third graph. This corresponds to the AVM [AGREEMENT [PERSON *1st*]]. Note that once more attributes correspond to transition labels, and further note that complex values (here [PERSON *1st*]) correspond to generated subgraphs. Next consider the fourth graph. This corresponds to the AVM

$$\begin{bmatrix} \text{AGREEMENT} & [\text{NUMBER } \textit{plural}] \\ \text{CASE} & \textit{nominative} \end{bmatrix}$$

As this example indicates, AVMs built using the fourth rule of AVM construction correspond to multiple branching nodes, with each attribute corresponding to a different branch label. Finally, the fifth graph is the result of unifying the third and fourth graph; clearly it's the natural merger of its two constituents.

For reasons that will shortly be clear, it is this idea of thinking about attributes and values in terms of labeled decorated graphs that has come to be seen as fundamental, and a number of definitions of Attribute Value

Structure have been given that attempt to pin it down. I'm now going to present my own definition. It's a particularly simple one, and as we shall see it has other virtues.

Let $\mathcal{L}$ and $\mathcal{A}$ be non-empty sets. We think of these as the set of *labels* and the set of *atomic information* respectively, and in what follows we assume that both $\mathcal{L}$ and $\mathcal{A}$ are either finite or countably infinite. An Attribute Value Structure (AVS) of signature $\langle \mathcal{L}, \mathcal{A} \rangle$ is a triple $\langle N, \{R_l\}_{l \in \mathcal{L}}, \{Q_\alpha\}_{\alpha \in \mathcal{A}} \rangle$, where N is a non-empty set called the set of nodes; for all $l \in \mathcal{L}$, R_l is a binary relation on N that is a partial function; and for all $\alpha \in \mathcal{A}$, Q_α is unary relation on N.

This is the concept of Attribute Value Structure we will work with throughout the paper. Let's consider a concrete example. Suppose while working with some linguistic theory we specify a signature $\langle \mathcal{L}, \mathcal{A} \rangle$ where $\mathcal{L}$ includes such items as PERSON, NUMBER, CASE, AGREEMENT and so on, and $\mathcal{A}$ includes such items as *3rd*, *2nd*, *1st*, *plural* and *nominative*. Then any graph whose decorated nodes are all decorated by items drawn from $\mathcal{A}$, and whose transitions are all uniquely labeled by items drawn from $\mathcal{L}$, with no two distinct transitions from the same node labeled the same way, is an AVS of such a signature. The five graphs given earlier are examples of such AVSs.

The definition subsumes the better known definitions of Attribute Value Structures found in the literature, and in particular those of Gazdar *et al* [10] [9], Johnson [12], and Kasper and Rounds [14]. Moreover it's not too liberal. There are only two reasonably common further restrictions on the binary relations that it does not impose. The first of these is *acyclicity*, which means that it is never possible to return to a node n by following some sequence of R_l transitions from n. The second is that AVSs must be *point generated*. That is, there is always some starting node $n_0 \in N$ such that every distinct node $n \in N$ is reachable via some sequence of transitions from n_0. Now both the classes of acyclic AVSs and point generated AVSs are of interest in their own right, and they'll both crop up in the work that follows, but neither the concept of acyclicity nor that of point generation seems to intrinsically belong to the notion of partial functional dependencies between attributes and values — at most they seem to be optional extras — and so we follow what seems to be the prevailing trend and don't insist upon them.

There are also three constraints computational linguists commonly place on node decoration which my definition ignores. The constraints are these. First, for all $n \in N$ and all $\alpha, \beta \in \mathcal{A}$, if $n \in Q_\alpha$ and $\alpha \neq \beta$ then $n \notin Q_\beta$. That is, no node can be decorated with distinct pieces of atomic information;

'constant-constant clashes' are forbidden. Second, for all $n \in N$, n is in Q_α for some $\alpha \in \mathcal{A}$ iff n is a terminal node. In short, this is the graphical analog of the edict forbidding 'constant-compound clashes'. Third, for all $n, n' \in N$, if $n \in Q_\alpha$ and $n' \in Q_\alpha$ then $n = n'$.

The main reason for ignoring these demands is that by doing so we accommodate a generalisation that is beginning to emerge in computational linguistics, namely the use of *sorts*[19]. Sorts are essentially pieces of atomic information that need not obey these three restrictions. It seems sensible to define basic notions in a manner that covers such natural generalisations, and this my definition does. Nonetheless, because the three restrictions listed above are still quite common, we shall pay particular attention to them in what follows, and I will call any AVS satisfying them a *naive* AVS. Note that the five graphs given earlier are all naive AVSs.

Having defined AVSs, it might seem natural to immediately define the unification of such structures. In fact I'm not going to do this at all. Following recent practice we are (eventually) going to consider unification not as a partial operation on AVSs, but as a partial operation on AVS *descriptions*. A short historical interlude is needed to motivate this.

Up until this point in the essay we've treated AVMs and AVSs as if they were pretty much the same thing. We've seen that AVMs give rise to graphs — indeed that each AVM can simply be regarded as a graph — and that AVSs are just the natural generalisation of this class of graphs. In short, I've implicitly treated the relationship between the two kinds of structure as if it were that of identity. I've presented the relationship in this way because it reflects a historical fact: working computational linguists did tend to view matters thus. However in the 1980s this view led to difficulties.

The problems were first observed when computational linguists began extending AVM style formalisms.[1] One extension was to write AVMs expressing *disjunctive* partial dependencies. The following AVM is a typical example:

$$\begin{bmatrix} \text{NUMBER} & \textit{plural} \\ \text{CASE} & \{\textit{nominative, genitive, accusative}\} \end{bmatrix}$$

In this AVM the attribute CASE takes one of the values *nominative*, *genitive*, or *accusative*, but it's not specified which. Another extension was to state *negative* partial dependencies:

[1] The following examples are taken from [13].

$$\begin{bmatrix} \text{NUMBER} & \textit{plural} \\ \text{CASE} & [-\ \textit{dative}] \end{bmatrix}$$

In this AVM it's specified that CASE does *not* take the value *dative*. Now, it's certainly natural to demand full Boolean expressivity, and arguably it's linguistically useful to do so — but even this relatively modest demand led to conceptual difficulties: the 'AVMs are graphs' intuition no longer seemed tenable. The point is this. Computational linguists tend to think in very concrete terms about AVMs and AVSs. An AVM simply *is* a graph, a graph that can be manipulated by a parser. (And of course, given what is going on in the computer, this can be a very helpful intuition.) But now consider what happens in the extended formalisms. The concrete graphs simply vanish. If negated AVMs are some sort of graph, then presumably they are a 'negated graph'. But what is a 'negated graph'? And what is a 'disjunctive graph'? Whatever they might be they're certainly not the stuff of robust intuitions. In short, natural demands of expressivity led to conceptual difficulties.

The solution to these problems was to draw a distinction that is a stock-in-trade of logic: it was to regard AVMs (and kindred formalisms) as languages for talking about labeled decorated graphs, not as the graphs themselves. That is, AVMs came to be regarded as syntactic entities, with AVSs supplying their interpretation: the relation of identity gave way to that of satisfiability. Note how the conceptual difficulties dissolve once this distinction is drawn: no longer does one have to think about a negative AVM as being some kind of bizarre 'negative graph': it's simply a way of denoting all those decorated graphs on which the condition in question does not obtain.[2]

This insight rapidly led to a stream of papers on the semantics of unification grammar formalisms and associated logical matters. New languages for talking about AVSs were devised, and attempts were made to find appropriate first order languages of AVSs. This paper belongs to that tradition. What makes it rather unusual is that it claims a privileged role for modal languages.

Note that AVSs as I have defined them are Kripke models. Consider the definition. If $\langle N, \{R_l\}_{l\in\mathcal{L}}, \{Q_\alpha\}_{\alpha\in\mathcal{A}}\rangle$ is an AVS then the first two components $\langle N, \{R_l\}_{l\in\mathcal{L}}\rangle$ are an $\mathcal{L}$ indexed multiframe, while $\{Q_\alpha\}_{\alpha\in\mathcal{A}}$ supplies the interpretation of propositional variables. In short, a modal language

[2] Pereira and Shieber [18] seem to have been the first to note the usefulness of this distinction in the study of unification formalisms. The later work of Kasper and Rounds [14] was also very influential.

with a $\mathcal{L}$ indexed collection of modalities and an $\mathcal{A}$ indexed collection of propositional variables is a suitable language for describing AVSs. Now this is a very trivial claim: of course modal languages can describe Kripke structures! Moreover it seems a rather dull claim to make: there's a huge choice of languages with which we can describe these AVSs — first order languages, second order languages, infinitary languages — so why all the fuss about finding yet another? The interest rises from the following observation: *Unbeknownst to themselves, computational linguists have been using propositional modal logic all along. Whether they work with ordinary AVMs, AVMs augmented with boxlabels, or with the path equations of PATR-II, they have been describing linguistic structure in modal languages, and their unification algorithms are essentially modal satisfaction algorithms.* The rest of the paper is devoted to justifying this claim and exploring the various languages concerned.

2 Modal logic

The language L of signature $\langle \mathcal{L}, \mathcal{A} \rangle$ has the following alphabet. First, it contains some $\mathcal{A}$ indexed set of symbols. These symbols are called *propositional variables*, though we sometimes call them pieces of atomic information, or something similar. In stating some definitions and theorems it can be useful to write propositional variables with their subscripts explicitly displayed, and when doing this we use the notation p_α, where $\alpha \in \mathcal{A}$. However whenever possible we suppress the subscripts, and when we do this we typically represent our propositional variables by p, q, and r. Second, we have at our disposal some $\mathcal{L}$ indexed set of modal operators; this set is assumed to be disjoint from the set of propositional variables. We typically write the elements of $\mathcal{L}$ as f, g, h and so on, thus our modal operators usually look like $\langle f \rangle$, $\langle g \rangle$, $\langle h \rangle$ etc. Finally we have at our disposal some truth functionally adequate collection of Boolean connectives (in what follows we choose $\neg$ and $\wedge$), and the punctuation symbols) and (.

We build the well formed formulas (wffs) or sentences of L out of this stock of symbols as follows. First, all propositional variables are wffs. Second, if ϕ and ψ are wffs then so are $(\neg\phi)$ and $(\phi \wedge \psi)$. Third, if $\langle f \rangle$ is a modality and ϕ is a wff then $(\langle f \rangle \phi)$ is a wff. Finally, nothing is a wff unless it can be constructed using only the previous three rules. When writing wffs we make free use of standard logical conventions. In particular, for all $l \in \mathcal{L}$ we use the notation $[l]\phi$ for $\neg\langle l \rangle\neg\phi$; we define $\vee$, $\rightarrow$, $\leftrightarrow$, $\bot$ and $\top$ in

the usual manner; and we drop brackets in accordance with the standard conventions.

As has already been noted, AVSs are simply Kripke models. Suppose we have fixed some signature $\langle \mathcal{L}, \mathcal{A} \rangle$. Let $\mathbf{M} = \langle N, \{R_l\}_{l \in \mathcal{L}}, \{Q_\alpha\}_{\alpha \in \mathcal{A}} \rangle$ be any AVS of this signature. Then we can interpret the language L (of the same signature) on $\mathbf{M}$ in the usual manner:

$$\begin{array}{l}
\mathbf{M} \models p_\alpha[n] \text{ iff } n \in Q_\alpha, \text{ for all } \alpha \in \mathcal{A} \\
\mathbf{M} \models \neg\phi[n] \text{ iff } \mathbf{M} \not\models \phi[n] \\
\mathbf{M} \models \phi \wedge \psi[n] \text{ iff } \mathbf{M} \models \phi[n] \ \& \ \mathbf{M} \models \psi[n] \\
\mathbf{M} \models \langle l \rangle \phi[n] \text{ iff } \exists n'(nR_l n' \ \& \ \mathbf{M} \models \phi[n']), \text{ for all } l \in \mathcal{L}
\end{array}$$

If $\mathbf{M} \models \phi[n]$ then we say that ϕ is true in the model $\mathbf{M}$ at n, or that $\mathbf{M}$ satisfies ϕ at n. If $\mathbf{N} = \langle N, \{R_l\}_{l \in \mathcal{L}} \rangle$ is a multiframe then an L model based on $\mathbf{N}$ is any AVS $\langle N, \{R_l\}_{l \in \mathcal{L}}, \{Q_\alpha\}_{\alpha \in \mathcal{A}} \rangle$. We say that a wff ϕ is valid on a multiframe $\mathbf{N}$ iff for all $n \in N$ and all models $\mathbf{M}$ based on $\mathbf{N}$, $\mathbf{M} \models \phi[n]$.

Let's now consider some concrete examples of the relationship between AVMs and wffs of L. In what follows we'll work in a language of signatures $\langle \mathcal{L}, \mathcal{A} \rangle$ where $\mathcal{L}$ contains such items as PERSON, AGREEMENT, CASE, and NUMBER, and $\mathcal{A}$ contains such items as *1st*, *plural*, *nominative*, *dative* and *genitive*. One notational convention will aid perspicuity: we'll represent the propositional variables by their subscripts. For example, instead of writing $p_{nominative}$ and p_{dative} we'll write *nominative* and *dative* respectively.

Unsurprisingly, the atomic AVM *1st* corresponds to the propositional variable *1st*. The AVM [PERSON *1st*] corresponds to the wff $\langle$PERSON$\rangle$*1st*. The AVM [AGREEMENT [PERSON *1st*]] corresponds to $\langle$AGREEMENT$\rangle\langle$PERSON$\rangle$*1st*. In short, attributes are modalities and values are wffs. Next consider the AVM

$$\left[\begin{array}{ll} \text{AGREEMENT} & [\text{PERSON } \textit{1st}] \\ \text{CASE} & \textit{nominative} \end{array}\right]$$

This corresponds to

$$\begin{array}{ll} & \langle \text{AGREEMENT} \rangle \langle \text{PERSON} \rangle \textit{1st} \\ \wedge & \langle \text{CASE} \rangle \textit{nominative} \end{array}$$

In short, vertical stacking corresponds to conjunction.

Now one could spell out formally the translation from AVM notation into an appropriate language of L, but as should be clear from these examples, this is an entirely trivial matter. The relationship between the two notations is not some baroque encoding, it is transparent: L allows a straightforward

linearisation of AVMs. Moreover this correspondence extends — just as trivially — to AVMs with full Boolean expressivity. For example corresponding to the disjunctive AVM

$$\begin{bmatrix} \text{NUMBER} & \textit{plural} \\ \text{CASE} & \{\textit{nominative, genitive, accusative}\} \end{bmatrix}$$

we have the wff

$$\langle \text{NUMBER} \rangle \textit{plural} \wedge \langle \text{CASE} \rangle (\textit{nominative} \vee \textit{genitive} \vee \textit{accusative});$$

while corresponding to the negative AVM

$$\begin{bmatrix} \text{NUMBER} & \textit{plural} \\ \text{CASE} & [- \textit{dative}] \end{bmatrix}$$

we have

$$\langle \text{NUMBER} \rangle \textit{plural} \wedge \langle \text{CASE} \rangle \neg \textit{dative}.$$

Let's turn next to the subject of unification. Once the distinction between AVSs and their descriptions has been drawn there is a very natural way to think about this. Intuitively unification is about combining the (partial) information we have about Attribute Value Structures, so let's say that two AVS descriptions are *unifiable* iff their combined information describes at least one possible Attribute Value Structure. Now, our AVS descriptions are L wffs, and in terms of L wffs this intuition cashes out as the following definition: *two L wffs ϕ and ψ are unifiable iff their conjunction $\phi \wedge \psi$ is satisfiable.*

This definition means that questions concerning L's satisfaction relation are the most pressing logical questions as far as computational linguistics is concerned. In the remainder of the section I'm going to sketch a number of basic results about this relation. First I'll give some completeness results, and then some finite model property results. Together these results show that satisfaction is decidable in the cases of interest to computational linguists. For L, of course, these results are rather obvious. What is important is that they will extend to the more powerful languages we'll consider later, thus showing that satisfaction — and hence unification — is decidable in these richer description languages. Unfortunately, however, it is not possible in the space available to discuss particular satisfaction algorithms, and I'll merely remark that perspicuous tableaux methods for building satisfying models exist. The complexity of the satisfiability problem for various

Attribute Value logics (including L, L^N and L^{KR}) has been analysed by Blackburn and Spaan [4].

The logics in L of interest in computational linguistics are rather simple to axiomatise.[3] Let's first deal with minimal AV logic, that is, the logic of the class of all AVSs. As axioms we take some complete axiomatisation of propositional calculus and in addition all instances of the following two modal schemas:

$$\begin{array}{ll} (K_\alpha) & [l](\phi \to \psi) \to ([l]\phi \to [l]\psi) \\ (Pfunc) & \langle l\rangle\phi \to [l]\phi, \end{array}$$

and as rules of proof we take modus ponens[4] and, for all $l \in \mathcal{L}$, the rule of necessitation.[5] Call this system K_{AV}.

K_{AV} is sound and strongly complete with respect to the class of all AV structures. Soundness is trivial. Completeness is not much harder. We can make a model for any K_{AV} consistent set of sentences Σ as follows. Consider the natural model on the Henkin multiframe $\mathbf{M}^H$.[6] The only reason it could fail to be an AV structure is if for some $l \in \mathcal{L}$, R_l^H was not partial functional. But, because K_{AV} has all instances of $\langle l\rangle\phi \to [l]\phi$ as axioms, it follows by standard modal reasoning that this is not possible, hence $\mathbf{M}^H$ is an AV structure. But for any consistent set of L wffs Σ, and any MCS Σ^∞ extending Σ, $\mathbf{M}^H \models \Sigma[\Sigma^\infty]$. (This is just an instance of the Fundamental Theorem concerning Henkin models.) Thus the strong completeness result follows.

This result can be improved. It is easy to see that K_{AV} is also complete with respect to the class of point generated AV structures. For, given a consistent set of sentences Σ for which we seek a model, extend it to an MCS Σ^∞, and then take the submodel of $\mathbf{M}^H$ generated by Σ^∞. Trivially this submodel is an AV structure; by definition it is point generated; and as modal truth transfers to generated submodels, this new model satisfies Σ at

[3] In the discussion that follows I assume familiarity with formal proofs, formal provability (here denoted by $\vdash$), maximal consistent sets of sentences (MCSs), and some very basic results of modal completeness theory. All the required material can be found in [11].

[4] From $\vdash \phi \to \psi$ and $\vdash \phi$ deduce $\vdash \psi$.

[5] From $\vdash \phi$ deduce $\vdash [l]\phi$.

[6] The Henkin multiframe (of signature $\langle \mathcal{L}, \mathcal{A}\rangle$) is the pair $\mathbf{H} = \langle H, \{R_l^H\}_{l\in\mathcal{L}}\rangle$, where H is the set of all maximal consistent sets of sentences in L; and for all $l \in \mathcal{L}$, and all h, $h' \in N$, $hR_l^H h'$ iff for all wffs ϕ, $[l]\phi \in h$ implies $\phi \in h'$. The natural model on the Henkin frame is $\langle \mathbf{H}, \{Q_\alpha\}_{\alpha\in\mathcal{A}}\rangle$, where for all $\alpha \in \mathcal{A}$, and all $h \in H$, we have that $h \in Q_\alpha$ iff $p_\alpha \in h$. We frequently call the natural model on the Henkin frame the Henkin model.

Σ^∞. So we have a new completeness result, and thus the logics in L of the class of all AV structures, and of point generated AV structures coincide.

In fact K_{AV} also axiomatises the class of acyclic AV structures. I'm not going to prove this here — it requires a more delicate model building process such as that we will use when dealing with Kasper Rounds logic — but it's worth noting why this is so. Essentially it reflects the fact that L is blind to some interesting aspects of relational structure; in particular it can't see cycles. Both of the stronger languages L^N and L^{KR} we shall later consider can see this structure, and in these languages the logic of acyclic frames does not coincide with the minimal logic.

Let's now consider what happens if we impose the three restrictions on the distribution of atomic information that many computational linguists make. First, if we want to forbid constant-constant clashes we add all instances of:

$$(Ccc) \quad p_\alpha \rightarrow \neg p_\beta, \text{ for all } \alpha, \beta \in \mathcal{A} \text{ such that } \alpha \neq \beta.$$

The addition of these axioms obviously rules out constant-constant clashes in the Henkin model, and thus we have another completeness result.

Second, to insist that atomic information is only instantiated at terminal nodes, we add as axioms all instances of:

$$(Term) \qquad p_\alpha \rightarrow \neg\langle l\rangle\top, \text{ for all } \alpha \in \mathcal{A}, \text{ and all } l \in \mathcal{L}.$$

Again the required completeness result is immediate.

Insisting that atomic information is instantiated at a unique node is rather more interesting. Let $\langle F\rangle$ and $\langle G\rangle$ be metavariables over sequences of primitive modalities, and v be a metavariable over propositional variables. Then we add all instances of the following schema:

$$(Nom_V) \quad \langle F\rangle(v \wedge \phi) \wedge \langle G\rangle(v \wedge \psi) \rightarrow \langle F\rangle(v \wedge \phi \wedge \psi)$$

We'll meet this schema again in the next section, and I'll defer discussing the intuition underlying it till then. Here I'll merely note that the effect its inclusion has on the Henkin model is to force each propositional variable to occur in at most one MCS in each point generated submodel. Hence by taking such a submodel we can verify any consistent set of sentences on a model satisfying the third constraint, and so we have another extended completeness result. These completeness results are all additive. In particular, note that the logic of naive attribute value structures is axiomatised by $K_{AV} + Ccc + Term + Nom$.

These logics all have the finite model property and are decidable. Establishing the finite model property is delightfully simple; here is a sketch of the argument. Suppose that some wff ϕ is satisfied in a model $\mathbf{M}$ at a point n. Then we can build a finite model $\mathbf{M}'$ out of $\mathbf{M}$ that also satisfies ϕ at n. We do this as follows. First, let the natural number k be the maximum depth of nesting of modalities in ϕ. Second, let $\mathcal{L}^\phi$ be the subset of $\mathcal{L}$ that contains precisely the indexes of those modalities that actually occur in ϕ. Note that $\mathcal{L}^\phi$ is finite. Now the key point to observe is that nodes more than k steps away from n, and nodes linked to n only by some sequence of steps involving a relation whose index in *not* in $\mathcal{L}^\phi$, are irrelevant to the truth value of ϕ. In short, we can chuck all these nodes out. So, let $\mathbf{M}'$ be the structure that arises by generating in $\mathbf{M}$ from n, but *only* generating on those relations indexed by $\mathcal{L}^\phi$, and *only* generating out k steps. I'll call this process *selective generation*. Now it's a standard exercise to show that $\mathbf{M}'$ is an AVS satisfying ϕ at n, but in fact we have something more: $\mathbf{M}'$ must also be finite. This follows from the fact that we're working with partial functional relations: any node can have at most one R_l successor for each $l \in \mathcal{L}$. Thus we have satisfied ϕ on a finite model. As all the classes of AVSs axiomatised above are closed under selective generation, the logics in L of these classes all have the finite model property. The decidability of these logics follows at once: the respective axiom systems give us a method of enumerating theorems, search through the appropriate finite models gives us a way of enumerating non-theorems, thus theoremhood in all these systems is a recursive concept.

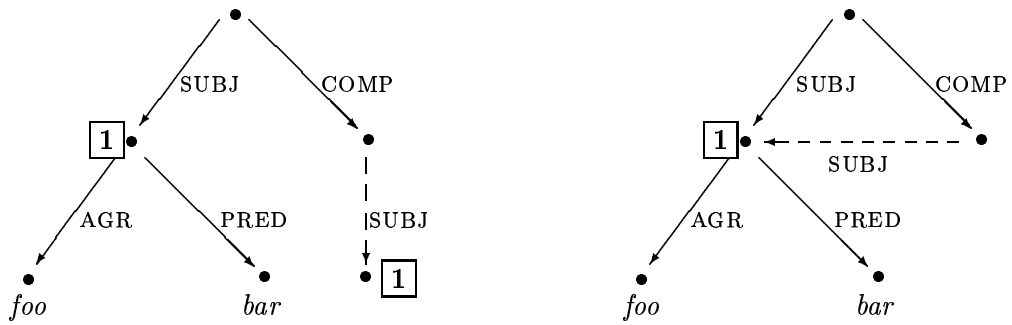
I'll close this section with some historical comments. Given the degree of current interest in Attribute Value logics, surprisingly few publications have even spotted the connection between AVM style notations and modal logic, and (apart from this paper) only a handful seem to take it seriously. The earliest papers I know of that deliberately use a modal language for dealing with natural language syntax are two papers due to Gazdar *et al* [10] [9]. In fact the language used is just L augmented by an extra modal operator $\Box$. Intriguingly, however, the authors seem to regard their language as modal only because of the presence of $\Box$: they don't seem to have made the connection between attributes and modalities outlined above. However their features are, syntactically, just modal operators; and the semantics they give to these features is just an ordinary Kripke semantics. The first statement of the link between attributes and modalities seems to have been made by Kracht [16]. In this elegant paper he makes the link explicit and then explores issues of completeness and decidability for Gazdar *et al's* language.

3 Modal logic with nominals

If there was no more to AVM notation than we have met so far, our investigation would be complete and we could conclude that computational linguists have been working with L all along. However there is a widely used aspect of AVM notation that hasn't yet been introduced: the use of *boxlabels*. Consider the following AVM:

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{AGR} & \textit{foo} \\ \text{PRED} & \textit{bar} \end{bmatrix} \boxed{1} \\ \text{COMP} & [\text{SUBJ} \quad \boxed{1}\,] \end{bmatrix}$$

The boxlabel is the entity $\boxed{1}$. Note that this occurs in two places. One of these occurrences is an ordinary value, but the other is something new: it is a superscript on the complex value taken by SUBJ. What this notation means (and what it does not mean) is explained by the following graphs:



The first graph is the result of treating $\boxed{1}$ simply as ordinary atomic information, but this is *not* what is intended. Rather, $\boxed{1}$ is a name: it must pick out precisely one node, and thus the required graph is the second one. The crucial difference is that as $\boxed{1}$ is a name, the dashed SUBJ transition is forced to *re-enter* the graph at the named node.

The ability to enforce re-entrancy plays an important role in the design of attribute value grammars. Moreover there is a clear computational intuition underlying re-entrancy: a notion of structure sharing has been introduced. If modal languages are to be taken seriously as languages of linguistic description we must find a way to deal with this aspect of AVMs. In fact this

can be done quite straightforwardly by making use of a *referential modal language*.

The key idea involved in making propositional modal languages referential can be traced back to early work by Arthur Prior and Robert Bull: it is to introduce a second sort of atomic symbol, symbols constrained to be true at exactly one node.[7] These new symbols — in this paper they are called *nominals* — in effect name the unique node they are true at. As we shall see, AVM boxlabels correspond to nominals.

Let's make this precise. We extend the language L (of signature $\langle \mathcal{L}, \mathcal{A} \rangle$) to the language with nominals L^N (of signature $\langle \mathcal{L}, \mathcal{A} \cup \mathcal{B} \rangle$) as follows. Augment the symbols of L with a denumerably infinite set of new symbols, and let these symbols be indexed by some set $\mathcal{B}$ distinct from both $\mathcal{L}$ and $\mathcal{A}$. These new symbols are called nominals, and we typically represent them by i, j, and k. The only addition we need make to the formation rules of L is to stipulate that all nominals are wffs; wffs are to be built up in the usual way from any mixture of nominals and variables.

We interpet L^N on AVSs of signature $\langle \mathcal{L}, \mathcal{A} \cup \mathcal{B} \rangle$ that satisfy the following requirement: for all $\alpha \in \mathcal{B}$, Q_α is a singleton. That is, we only interpret L^N on those AVSs where nominals act as names. If $\mathbf{M} = \langle N, \{R_l\}_{l \in \mathcal{L}}, \{Q_\alpha\}_{\alpha \in \mathcal{A} \cup \mathcal{B}} \rangle$ is an AVS satisfying this constraint then we say it is an L^N model. The truth definition for L^N is that for L augmented by the following clause:

$$\mathbf{M} \models i_\alpha[n] \text{ iff } n \in Q_\alpha, \text{ for all } \alpha \in \mathcal{B}.$$

Finally, we say that a wff ϕ of L^N is valid on an $\mathcal{L}$ indexed multiframe $\mathbf{N}$ iff for all L^N models $\mathbf{M} = \langle \mathbf{N}, \{Q_\alpha\}_{\alpha \in \mathcal{A} \cup \mathcal{B}} \rangle$ and all $n \in N$, $\mathbf{M} \models \phi[n]$.

Nominals correspond to boxlabels quite straightforwardly. Consider once more the following AVM:

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{AGR} & \textit{foo} \\ \text{PRED} & \textit{bar} \end{bmatrix} \boxed{1} \\ \text{COMP} & [\text{SUBJ} \quad \boxed{1}\,] \end{bmatrix}$$

This corresponds to the following L^N wff:

$$\begin{array}{ll} & \langle \text{SUBJ} \rangle (i \wedge \langle \text{AGR} \rangle \textit{foo} \wedge \langle \text{PRED} \rangle \textit{bar}) \\ \wedge & \langle \text{COMP} \rangle \langle \text{SUBJ} \rangle i \end{array}$$

[7] For Prior's work consult [20], for Bull's work [5]. For recent results on the subject see [7], [8], [2] or [3].

In short, the nominal i is doing the work that $\boxed{1}$ does in the AVM, and more generally, using a boxlabel superscript amounts to conjoining a nominal. Thus L^N allows a straightforward linearisation of those AVMs that utilise boxlabels, and indeed it seems natural to say that such AVMs simply are L^N wffs.

Quite a lot is known about modal languages with nominals; I'll sketch here some results that will prove useful. Moving from L to L^N has resulted in a genuine gain in expressive power. It's well known that ordinary modal languages (such as L) are blind to some simple aspects of frame structure. For example, there is no L wff which is valid on precisely the irreflexive multiframes; L can't define irreflexivity. Neither can L define antisymmetry, nor intransitivity. However all these conditions can be defined in L^N. It's a simple exercise to check that $i \rightarrow \neg\langle f\rangle i$ is valid on precisely those multiframes in which R_f is irreflexive; that $i \rightarrow [f](\langle f\rangle i \rightarrow i)$ is valid on precisely those multiframes in which R_f is antisymmetric; and that $\langle f\rangle\langle f\rangle i \rightarrow \neg\langle f\rangle i$ is valid on precisely those multiframes in which R_f is intransitive. Moreover L^N, unlike L, can 'see cycles', and thus can forbid their presence. Define $Acyc$ to be

$$\{i \rightarrow \neg\langle F\rangle i : \langle F\rangle \text{ is a non-null sequence of primitive modalities}\}.$$

It is easy to to show that for any multiframe $\mathbf{N}$, $\mathbf{N} \models Acyc$ iff $\mathbf{N}$ is acyclic.

This increase in expressive power has some immediate model theoretic consequences: taking p-morphic images and forming disjoint unions are not validity preserving operations (they are for L), and unraveling is no longer a method that will produce equivalent models. On the other hand, forming generated subframes is a validity preserving operation for L^N, and ultrafilter extension formation back preserves in the usual manner the validity of L^N wffs.

Axiomatising the L^N logics of the various classes of AVSs of interest in computational linguistics is routine. In fact these logics are straightforward extensions of the corresponding logics in L. A key role in these extensions is played by a schema called Nom_N, a nominal form of the Nom_V schema we met in the last section. Let's begin by considering the logic in L^N of the class of all AVSs. We axiomatise this minimal logic as follows. Take as axioms all L^N instances of the K_{AV} schemas and rules, and in addition, add as axioms all instances of the following schema:

$$(Nom_N) \quad \langle F\rangle(n \wedge \phi) \wedge \langle G\rangle(n \wedge \psi) \rightarrow \langle F\rangle(n \wedge \phi \wedge \psi)$$

In this schema both $\langle F \rangle$ and $\langle G \rangle$ are metavariables over sequences of primitive modalities, and n is a metavariable over nominals.

As I remarked in the last section, the effect the inclusion of such *Nom* style axioms has on Henkin models is to ensure that in any point generated submodel no nominal occurs in two distinct MCSs. This is the crucial technical point, but it's just as important to grasp the underlying intuition. To see what this is, consider the following instance of Nom_N:

$$\langle f \rangle \langle g \rangle \langle f \rangle \langle h \rangle (i \wedge p) \wedge \langle g \rangle \langle g \rangle (i \wedge q) \rightarrow \langle f \rangle \langle g \rangle \langle f \rangle \langle h \rangle (i \wedge p \wedge q).$$

This wff can be paraphrased as follows. Suppose we are at some node n in an AVS, and suppose that by making the series of transitions R_f followed by R_g, R_f and R_h we get to a node n_1 labelled by i and bearing the information p. Further, suppose that by making from n the transition sequence R_g followed by R_g we get to a node n_2 labelled by i and bearing the information q. *But since nominals label exactly one node, n_1 is just n_2. That is, both paths have taken us to the same node, thus this node must bear both the information p and the information q.*

$K_{AV} + Nom_N$ captures the minimal logic. The completeness result is straightforward. Given a consistent set of sentences Σ for which we seek a model, expand it to an MCS Σ^∞ and take the subframe of the Henkin multiframe generated by Σ^∞. Now, defining for all $\alpha \in \mathcal{A} \cup \mathcal{B}$ the Q_α in the natural way may not produce an L^N model, but in fact the resulting structure can only fail to be one for a rather trivial reason: there may be some nominals that are not true anywhere. It is easy to deal with such 'unused nominals': simply add a brand new node n^+ to the Henkin model, leave the relations unchanged, and let the unused nominals be true at this new node. The resulting structure is an L^N model that satisfies Σ at Σ^∞.

Improving this result is easy: $K_{AV} + Nom_N$ is also complete with respect to the point generated AV structures. To see this, follow the procedure just described but with one addition: choose some $f \in \mathcal{L}$ and insist that $n^+ R_f \Sigma^\infty$. It is clear that the resulting model is point generated by n^+ and thus we have our improved completeness result.

The L^N logic of the acyclic AVSs differs from the minimal logic. In fact this logic is axiomatised by $K_{AV} + Nom_N + Acyc$, where $Acyc$ is as defined above. I won't prove this here; as with the corresponding result for L a more sensitive model building procedure is required. The three conditions on atomic information distribution popular in computational linguistics are

captured by adding all L^N instances of *Ccc*, *Term*, and Nom_V respectively. As with L, these results are additive, thus the L^N logic of naive AVSs is axiomatised by $K_{AV} + Nom_N + Ccc + Term + Nom_V$. Using the method of selective generation discussed in the previous section, it is clear that all these logics have the finite model property. Thus by the usual argument all these logics are decidable.

There are only three papers which make use of modal languages with nominals for dealing with linguistic structure. Bird and Blackburn [1] use a modal language with nominals for talking about phonological information. The twist in this paper is that a number of different AVSs are pictured as being strung out along a time axis, and additional tense operators are introduced to cope with the temporal structure. That is, the language is an attempt to capture Attribute Value style reasoning and temporal reasoning in a single framework. As in L^N, the nominals are used to name nodes and enforce re-entrancy.

Reape [21] uses a language similar to L^N for talking about syntactic information. This language is then extended to a larger language containing polyadic modal operators to enable the *set valued features* and *functional dependencies* of such linguistic theories as HPSG to be dealt with. This is a significant achievement. Reape's work reveals the inherent simplicity and elegance of the ideas underlying HPSG, and shows that even this formalism hasn't outstripped the resources of simple propositional modal languages. There is an important difference, however, betweens L^N and Reape's basic language: L^N contains both propositional variables and nominals, whereas Reape's language only contains nominals. Not only are nominals used to force re-entrancy, they are also used to represent ordinary atomic information. This is odd. It means that not only is the naive interpretation of atomic information enforced, but that in addition, in every AVS every piece of atomic information must be realised somewhere. This last condition clearly violates the 'partial information' intuition underlying much work in Attribute Value grammar. Moreover, many reasonable looking structures are ruled out as AVSs: note, for example, that none of the graphs drawn in this paper are AVSs under this definition, for in none of them is the atomic information *3rd* realised anywhere.

Finally, a chapter of Ruhrberg's thesis [24] examines modal languages with nominals.

4 Kasper Rounds logic

As we have seen, multimodal languages with nominals correspond neatly to the AVM notation actually used by computational linguists. However AVM notation, though widely used, is by no means the only notation computational linguists use when working with AV structures: another important notation is that provided by the PATR-II system. In this section we'll briefly examine the PATR-II notation and abstract one of its key devices, its mechanism for enforcing re-entrancy. This mechanism is very different from the 'boxlabels' used in AVM notation. In fact the PATR-II mechanism doesn't work by labeling nodes at all. Rather, it's a mechanism for directly equating two sequences of transitions. We'll add this simple mechanism to L forming the language L^{KR}, the language of Kasper Rounds logic, and then answer some of the more obvious questions about it: has the new mechanism led to genuine expressive gains? How does it relate to L^N? Can we give sound and complete axiomatisations of the logics of the various classes of AV structures of interest to the computational linguist, and are these logics decidable?

Before proceeding, a historical note. As the term 'Kasper Rounds logic' indicates, this language (or rather a certain notational variant of it) has been discussed before. In fact the work of Kasper and Rounds [14] on this language did a great deal to arouse the interest computational linguists are currently showing in Attribute Value logics, for it made it very clear that the distinction logicians draw between a language and its semantics was a fruitful way of looking at puzzles involving disjunctive feature structures. However Kasper and Rounds proved relatively few results about L^{KR}. They give some near the end of [22], but the bulk of this paper is devoted to defining the relevant languages and proving a completeness result for the negation free fragment of L. Incidentally, most of the results Kasper and Rounds give are not obtained using modal techniques; although they spotted the similarity between modal operators and their labels, they don't seem to have attached any importance to it, and in particular don't make use of any of the standard tools and results of modal logic. In fact the only other logical investigation of L^{KR} seems to be that of Moss [17]. This paper takes an essentially (Kripke) model theoretic stance towards L^{KR} (and some other Attribute Value logics) and should be of interest to the readers of this paper.

So, what is PATR-II? Basically it's an 'implemented grammar formalism'. That is, it's a program which provides a high level interface language tailored to the needs of the working linguist, together with a parser. The linguist writes a grammar in the interface language and then hands it over

to the parser, which will cheerfully chew away at any sentence fed to it to see if it meets the grammatical specification. PATR-II is used by computational linguists to develop and test grammars.

Now, what is of interest for the present discussion is the high level interface language. The essential core of this is a notation for describing AV structures. The user specifies these by writing *path equations*. There are essentially two types of path equations the user can write, and we'll consider each in turn.

The first type of path equation equates a list of attributes with a value. For example, a user may write:

$$\langle \text{VP VERB HEAD NUM} \rangle = \textit{sing}.$$

The item between the angle brackets are attributes, and the item on the right hand side of the equality symbol is an atomic value. The meaning of this first type of path equation is that by making the sequence of transitions encoded by the list on the left, one will arrive at at a node bearing the atomic information *sing*. In short, this path equation bears the same information as the L wff

$$\langle \text{VP} \rangle \langle \text{VERB} \rangle \langle \text{HEAD} \rangle \langle \text{NUM} \rangle \textit{sing},$$

and more generally, this first type of path equation can be represented by means of L wffs.

It is in the second type of path equation that we meet something new. In this second type of equation two lists of attributes are equated. For example, the user might specify that

$$\langle \text{VP HEAD} \rangle = \langle \text{VP VERB HEAD} \rangle.$$

What this specification means is that making the sequence of transitions encoded by the list on the left takes one to the same node as if one had made the sequence of transitions encoded by the list on the right. Briefly, both transition sequences lead to the same node. Thus the second type of path equation allows the user to specify re-entrancy.

This specification mechanism certainly looks different from anything we've seen before; can we tease it apart from its PATR-II setting and add it to L? We can, and rather easily. First, as we've already seen from our discussion of the first type of path equation, PATR-II's attribute lists correspond to sequences of L modalities. So, in terms of the syntax of L, what the second type of path equation amounts to is a *mechanism for equating*

two sequences of modalities. Let's enrich L by adding a new primitive, a primitive we will write as $\approx$. The purpose of this new symbol will be to allow a new kind of wff to be formed, wffs such as

$$\langle \text{VP} \rangle \langle \text{HEAD} \rangle \approx \langle \text{VP} \rangle \langle \text{VERB} \rangle \langle \text{HEAD} \rangle,$$

and we will define the semantics of these new wffs so that they do capture the meaning of the second sort of PATR-II path equation.

It's important to note that this is the only way we're going to use our new symbol $\approx$. We're *not* going to use it to encode the first type of PATR-II path equation. For example we *won't* represent $\langle \text{VP VERB HEAD NUM} \rangle = \mathit{sing}$, by $\langle \text{VP VERB HEAD NUM} \rangle \approx \mathit{sing}$, in fact this last sequence of symbols won't even be a well formed expression of our enriched language. Nor do we need such expressions; as we've already seen we can capture the meaning of the PATR-II expression by means of $\langle \text{VP} \rangle \langle \text{VERB} \rangle \langle \text{HEAD} \rangle \langle \text{NUM} \rangle \mathit{sing}$. Thus the new device will only be used to mimic the second type of PATR-II path equation, and from now on whenever we talk of 'path equations' we'll mean equations of this type.

One final matter before we turn to the details. In addition to $\approx$ we're going to add a second new primitive symbol, 0. This will act as a name for the null transition. Having this symbol will enable us to write such path equations as $\langle g \rangle \langle f \rangle \approx 0$ and $0 \approx \langle g \rangle \langle f \rangle \langle g \rangle$. For example the first equation means that making an R_g transition followed by an R_f transition is the same as making the null transition. That is, the path $R_g R_f$ terminates at its starting point. Similarly, the second equation means that the path $R_g R_f R_g$ terminates at its starting point.

Enough motivation, let's define the language L^{KR} of signature $\langle \mathcal{L}, \mathcal{A} \rangle$. We take as primitive symbols all the symbols of L (of signature $\langle \mathcal{L}, \mathcal{A} \rangle$), and in addition two new symbols $\approx$ and 0. That is, we have at our disposal some $\mathcal{A}$ indexed set of propositional variables, some $\mathcal{L}$ indexed set of modalities, the Boolean connectives, brackets, $\approx$ and 0. As a first step towards defining the L^{KR} wffs we define the set of *path equations* of L^{KR}. So let $\langle f_\alpha \rangle \cdots \langle f_\beta \rangle$ and $\langle f_\gamma \rangle \cdots \langle f_\delta \rangle$ be non-null sequences of L^{KR} modalities.Then $0 \approx 0$, $0 \approx \langle f_\alpha \rangle \cdots \langle f_\beta \rangle$, $\langle f_\alpha \rangle \cdots \langle f_\beta \rangle \approx 0$ and $\langle f_\alpha \rangle \cdots \langle f_\beta \rangle \approx \langle f_\gamma \rangle \cdots \langle f_\delta \rangle$ are all and only the path equations of L^{KR}. With this notion to hand we can now define the L^{KR} wffs. Firstly, all propositional variables are wffs. Secondly, all path equations are wffs. Lastly, the set of wffs is closed under the Boolean operations and the application of modal operators. In short, we have a language that is syntactically like L save that as well as the ordinary atomic

symbols we have (so to speak) the 'composite atomic' symbols that are the path equations.

Suppose that we've fixed the signature we're working with. Then the models for L^{KR} are just the ordinary L models of that signature. If $\mathbf{M}$ is an L^{KR} model we interpret L^{KR} wffs in $\mathbf{M}$ as follows. Firstly, propositional variables, Boolean combinations and modal applications are interpreted just as before. As for the path equations, these are interpreted as follows:

$\mathbf{M} \models 0 \approx 0[n]$ for all $n \in N$
$\mathbf{M} \models 0 \approx \langle f_\alpha \rangle \cdots \langle f_\beta \rangle [n]$ iff $nR_\alpha \ldots R_\beta n$
$\mathbf{M} \models \langle f_\alpha \rangle \cdots \langle f_\beta \rangle \approx 0[n]$ iff $nR_\alpha \ldots R_\beta n$
$\mathbf{M} \models \langle f_\alpha \rangle \cdots \langle f_\beta \rangle \approx \langle f_\gamma \rangle \cdots \langle f_\delta \rangle [n]$ iff $\exists n'(nR_\alpha \ldots R_\beta n' \ \& \ nR_\gamma \ldots R_\delta n')$

We say that an L^{KR} wff ϕ is true in $\mathbf{M}$ at a node n iff $\mathbf{M} \models \phi[n]$. Validity is defined as for L.

It is important that the reader understands the shorthand used in the above clauses. For example, $nR_\alpha \ldots R_\beta n$ is really an abbreviation of

$$\exists n_1, \ldots, n_k (nR_\alpha n_1 \ \& \cdots \& \ n_k R_\beta n),$$

while $\exists n'(nR_\alpha \ldots R_\beta n' \ \& \ nR_\gamma \ldots R_\delta n')$ is short for

$$\exists n', n_1, \ldots, n_k, m_1, \ldots, m_j (nR_\alpha n_1 \ \& \cdots \& \ n_k R_\beta n' \ \& \ nR_\gamma m_1 \ \& \cdots \ \& \ m_j R_\delta n').$$

That is, all but the first clause contains a statement concerning path existence. Note, by the way, that the truth definition makes sense even if we drop the requirement that the R_l are partial functional. In fact using L^{KR} on non partial functional structures may be useful for various other applications (for example in phonology or knowledge representation), though a discussion of this point is not possible here.

L^{KR} is not quite the language Kasper and Rounds discuss. There are two differences. First, their language lacks negation. Second, their notation for enforcing path equalities is rather different. Their language has a device $[\![\ ]\!]$ which combines with k lists of labels (for any $k \geq 1$)) to assert that the transition sequences named by the k lists are coterminous. For example, $[\![\langle f \rangle, \langle gh \rangle, \langle gg \rangle]\!]$ is true at a node n iff the three paths R_f, $R_g R_h$ and $R_g R_g$ all lead from n to the same node. In L^{KR} we would express this particular path equality by means of $\langle f \rangle \approx \langle g \rangle \langle h \rangle \wedge \langle g \rangle \langle h \rangle \approx \langle g \rangle \langle g \rangle$, and more generally the expressive equivalence of the two notations is transparent. In spite of

these differences I'll continue to call L^{KR} the language of Kasper Rounds logic.

It's also worth remarking that there is another way to formulate L^{KR}, namely to treat 0 as a 'dummy modality'. To be more precise, instead of introducing 0 in the manner we did above, we could instead introduce it as a special modality $\langle 0 \rangle$. The advantage of this variant — let's call it $L^{KR'}$ — is that it makes possible a simpler statement of the syntax and semantics of path equations. First we define the modalities of $L^{KR'}$ to be the usual $\mathcal{L}$ indexed modalities together with $\langle 0 \rangle$, and then we state that if $\langle f_\alpha \rangle \cdots \langle f_\beta \rangle$ and $\langle f_\gamma \rangle \cdots \langle f_\delta \rangle$ are non-null sequences of $L^{KR'}$ modalities, then $\langle f_\alpha \rangle \cdots \langle f_\beta \rangle \approx \langle f_\gamma \rangle \cdots \langle f_\delta \rangle$ is a path equation. In short, we have only one type of path equation instead of the four of L^{KR}. Next, we add to the formation rules of L the following two clauses: all path equations are wffs; and if ϕ is a wff then so is $\langle 0 \rangle \phi$. To interpet $L^{KR'}$ we associate $\langle 0 \rangle$ with the identity relation on nodes, that is, with the 'null transition'. So instead of having to add four semantic clauses for path equations, we need only the following:

$$\mathbf{M} \models \langle f_\alpha \rangle \cdots \langle f_\beta \rangle \approx \langle f_\gamma \rangle \cdots \langle f_\delta \rangle [n] \text{ iff } \exists n' (n R_\alpha \ldots R_\beta n' \ \& \ n R_\gamma \ldots R_\delta n').$$

Note that because $\langle 0 \rangle$ is associated with the identity relation on nodes, the following holds:

$$\mathbf{M} \models \langle 0 \rangle \phi [t] \ \text{ iff } \ \mathbf{M} \models \phi [t].$$

It is clear that L^{KR} and $L^{KR'}$ are expressively equivalent — indeed L^{KR} is essentially a sublanguage of $L^{KR'}$ in which all the path equations are in a certain normal form. In this paper we work with L^{KR} because, for the purposes of this paper, it is slightly simpler: although the initial definitions are more cumbersome we avoid some rather dull work when proving completeness. For other purposes $L^{KR'}$ might well be the more natural choice.

Let's begin our logical examination of L^{KR}. This language is indeed more expressive than L: using path equations we can define conditions on multiframes not definable in L. For example, it is straightforward to verify that for any multiframe $\mathbf{N} = \langle N, \{R_l\}_{l \in \mathcal{L}} \rangle$ and any $f \in \mathcal{L}$, $\mathbf{N} \models \neg(0 \approx \langle f \rangle)$ iff R_f is irreflexive; $\mathbf{N} \models \langle f \rangle \langle f \rangle \approx 0 \rightarrow \langle f \rangle \approx 0$ iff R_f is antisymmetric; and $\mathbf{N} \models \neg(\langle f \rangle \approx \langle f \rangle \langle f \rangle)$ iff R_f is intransitive. Note further that in L^{KR} we can pin down the concept of acyclicity, something that cannot be done in L. Define *Acyc* to be:

$$\{\neg(0 \approx \langle F \rangle) : \langle F \rangle \text{ is a non-null sequence of primitive modalities}\}.$$

Then for any multiframe $\mathbf{N}$, $\mathbf{N} \models \mathit{Acyc}$ iff $\mathbf{N}$ is acyclic. The proof is straightforward.

These results have some immediate (negative) model theoretic corollaries. Because we can define intransitivity, unraveling is not a truth preserving operation on models. Because we can define irreflexivity, p-morphisms do not preserve frame validity. On the other hand, the following positive model theoretic results are more or less immediate. The forming of generated subframes and disjoint unions are not only truth preserving operations on models, they are also validity preserving operations on frames. Moreover one can give a natural definition of a validity preserving morphism for L^{KR} — essentially one strengthens the backwards clause of the p-morphism definition to fit the requirements of $\approx$.

What is the relationship between L^{KR} and the nominal language L^N? Let's compare the respective powers of the two languages for defining conditions on multiframes.

First, L^N is not weaker than or equal to L^{KR} in such expressive power. To see this, note that L^N is strong enough to define the universal relation: for any multiframe $\mathbf{N}$, $\mathbf{N} \models \langle f \rangle i$ iff $R_f = N \times N$. However L^{KR} can't define this condition, for disjointly sticking together two multiframes results in a multiframe that is not universally related, and L^{KR} validity is preserved under disjoint union formation. Thus there is at least one condition on multiframes definable in L^N that is not definable using L^{KR}.

However we also have that L^{KR} is not weaker than or equal to L^N in expressive power. This result hinges on the following observation: L^{KR} validity is *not* back preserved under the formation of ultrafilter extensions. To see this, note that in L^{KR} we can define the class of multiframes such that every node n f-precedes a node n' that is f-reflexive (that is, $\forall n \exists n'(nR_f n' \wedge n'R_f n')$), for $\langle f \rangle(\langle f \rangle \approx 0)$ picks out precisely such multiframes. Now, the ultrafilter extension of the natural numbers in their usual order satisfies this condition, but clearly the natural numbers themselves don't, thus the familiar ultrafilter preservation result doesn't hold for L^{KR}. But we've already noted that the result *does* hold for L^N, so L^N cannot define $\forall n \exists n'(nR_f n' \wedge n'R_f n')$. Thus there is at least one condition on multiframes definable in L^{KR} that is not definable using L^N. In short, the two languages are incomparable in expressive power if this is measured in terms of multiframe defining power.

It's worth noting, however, that L^N can simulate all path equations and negations of path equations. First, we can simulate path equations of the form $0 \approx \langle F \rangle$, (or equivalently, $\langle F \rangle \approx 0$), where $\langle F \rangle$ is a sequence of

primitive modalities. It is straightforward to verify that for any multiframe $\mathbf{N}$:

$$\mathbf{N} \models 0 \approx \langle F \rangle \text{ iff } \mathbf{N} \models i \rightarrow \langle F \rangle i.$$

Second, L^N can simulate the negations of all such path equations. For any multiframe $\mathbf{N}$ we have that:

$$\mathbf{N} \models \neg(0 \approx \langle F \rangle) \text{ iff } \mathbf{N} \models i \rightarrow \neg\langle F \rangle i.$$

Third, L^N can simulate the effect of *negations* of path equations of the form $\langle F \rangle \approx \langle G \rangle$ where both $\langle F \rangle$ and $\langle G \rangle$ are sequences of primitive modalities, for it is easy to show that for all multiframes $\mathbf{N}$:

$$\mathbf{N} \models \neg(\langle F \rangle \approx \langle G \rangle) \text{ iff } \mathbf{N} \models (\langle F \rangle \top \wedge \langle G \rangle \top) \rightarrow (\langle F \rangle i \rightarrow \neg\langle G \rangle i).$$

However matters appear to be more subtle when it comes to path equations of the form $\langle F \rangle \approx \langle G \rangle$. It seems to make a difference whether our working restriction to partial functional multiframes is in force or not. If it is — that is, if we are only considering those multiframes that could form the relational skeleton of an AVS — then we *can* define such path equations. For any partial functional multiframe $\mathbf{N}$ we have that

$$\mathbf{N} \models \langle F \rangle \approx \langle G \rangle \text{ iff } \mathbf{M} \models (\langle F \rangle i \leftrightarrow \langle G \rangle i) \wedge F\top \wedge \langle G \rangle \top.$$

On the other hand, if we drop this working restriction and consider the class of all multiframes $\mathcal{U}$, it doesn't seem that L^N can define all the conditions that L^{KR} can. Let's consider a concrete L^{KR} wff, say $\langle f \rangle \approx \langle g \rangle$. This is valid on a multiframe $\mathbf{N}$ iff for all $n \in N$ there is an $n' \in \mathbf{N}$ such that $nR_f n'$ and $nR_g n'$. That is, from every node there is a node accessible by both R_f and R_g transitions. It doesn't seem that L^N can define this condition over $\mathcal{U}$. An obvious attempt to do so is $\langle f \rangle i \rightarrow \langle g \rangle i$, but this doesn't work: while this wff is valid on all multiframes satisfying the condition, it's not valid on only such multiframes. For example, it's valid on all multiframes whose R_f relation is empty. A second attempt might be to use the wff that defines this condition on the class of all partial functional frames, namely $(\langle f \rangle i \leftrightarrow \langle g \rangle i) \wedge \langle f \rangle \top \wedge \langle g \rangle \top$, but as a simple check shows, this doesn't work either. Now, I don't believe that this condition is definable over $\mathcal{U}$ by any L^N wff, but I don't know how to prove it. This would be an interesting matter to investigate further.[8]

[8] As the referee of this paper pointed out, if we work in a *tensed* version of L^N — that

There are other differences between L^{KR} and L^N. For example, let $\mathcal{U}$ be the class of all multiframes. Then $\mathcal{U} \models [f_\alpha]\phi_1 \vee \ldots \vee [f_\beta]\phi_n$ iff $\mathcal{U} \models \phi_k$ for some $k, (1 \leq k \leq n)$, where all the ϕ_k are wffs of L^{KR}. That is, L^{KR} has the disjunction property. On the other hand, it's straightforward to show that this cannot be the case for L^N. The model theoretic fact underlying these observations is that whereas when we 'root' two or more L^{KR} models we obtain another L^{KR} models, rooting even two L^N models is not a process guaranteed to yield an L^N model.[9]

Handling L^{KR} proof theoretically takes rather more effort than with L or L^N. The basic result, which will take us some pages to establish, is as follows: the minimal logic of AV structures in L^{KR} is axiomatised by adding as axioms all wffs which are instances of the following schemas to K_{AV}:

KR1	$0 \approx 0$
KR2	$\langle F\rangle \approx \langle G\rangle \rightarrow \langle G\rangle \approx \langle F\rangle$
KR3	$\langle F\rangle \approx \langle G\rangle \wedge \langle G\rangle \approx \langle H\rangle \rightarrow \langle F\rangle \approx \langle H\rangle$
KR4	$\langle F\rangle\phi \rightarrow \langle F\rangle \approx \langle F\rangle$
KR5	$\langle F\rangle \approx \langle G\rangle \rightarrow \langle F\rangle\top$
KR6	$\langle F\rangle(\langle G\rangle \approx \langle H\rangle) \leftrightarrow \langle F\rangle\langle G\rangle \approx \langle F\rangle\langle H\rangle$
KR7	$\langle F\rangle \approx \langle G\rangle \wedge \langle F\rangle\phi \rightarrow \langle G\rangle\phi$
	$\langle F\rangle \approx 0 \wedge \langle F\rangle\phi \rightarrow \phi$
	$0 \approx \langle G\rangle \wedge \phi \rightarrow \langle G\rangle\phi$
KR8	$\langle F\rangle \approx \langle G\rangle \wedge \langle F\rangle\langle H\rangle\phi \rightarrow \langle F\rangle\langle H\rangle \approx \langle G\rangle\langle H\rangle$
	$\langle F\rangle \approx 0 \wedge \langle F\rangle\langle H\rangle\phi \rightarrow \langle F\rangle\langle H\rangle \approx \langle H\rangle$
	$0 \approx \langle G\rangle \wedge \langle H\rangle\phi \rightarrow \langle H\rangle \approx \langle G\rangle\langle H\rangle$
KR9	$\langle F\rangle \approx \langle E\rangle \wedge \langle E\rangle\langle G\rangle \approx \langle H\rangle \rightarrow \langle F\rangle\langle G\rangle \approx \langle H\rangle$
	$\langle F\rangle \approx 0 \wedge \langle G\rangle \approx \langle H\rangle \rightarrow \langle F\rangle\langle G\rangle \approx \langle H\rangle$
	$0 \approx \langle E\rangle \wedge \langle E\rangle\langle G\rangle \approx \langle H\rangle \rightarrow \langle G\rangle \approx \langle H\rangle$

In these schemas ϕ is a metavariable over arbitrary wffs and $\langle E\rangle$, $\langle F\rangle$, $\langle G\rangle$ and $\langle H\rangle$ are metavariables over sequences made up from modalities and 0. Note that the schemas divide into two groups. Instances of KR1, KR2,

is, in a language in which for each forward looking modality $\langle f\rangle$ there is a corresponding backward looking modality $\langle f\rangle^{-1}$ — then we can express $\langle f\rangle \approx \langle g\rangle$ over $\mathcal{U}$ by means of $i \rightarrow \langle f\rangle\langle g\rangle^{-1}i$. On the other hand, even with this extension it doesn't seem that $\langle f\rangle(\langle f\rangle \approx \langle g\rangle)$ is definable.

[9] Briefly, given a non-empty set of models $\{\mathbf{M}_k : k \in \mathcal{K}\}$, we root this collection by taking its disjoint union, and then adding a new node n_∞ such that $n_\infty R_l n$, for all n in the disjoint union, but for any such n it is not the case that $nR_l n_\infty$. For a general discussion of rooting and the disjunction property see [11, pages 96–100]; for a discussion of these topics in languages with nominals see [3, pages 67 – 68] or [2, page 11].

KR4, KR5 and of the left to right implication in KR6 would be valid even if we dropped the partial functional restriction we have imposed on AVSs, whereas instances of KR3, KR7–KR9 and the right to left implication in KR6 depend on this restriction for their validity.

Proving soundness for the resulting system is straightforward. But how can we prove completeness? For a start, note that merely taking a generated subframe of the Henkin multiframe is *not* going to yield a model. To see what goes wrong consider a simple path equation, say $\langle f \rangle \approx \langle g \rangle$. For the Henkin method to work in the familiar straightforward manner we would certainly have to be able to prove that for any node h in the Henkin model $\mathbf{M}^H$ we have that:

$$\langle f \rangle \approx \langle g \rangle \in h \text{ iff } \mathbf{M}^H \models \langle f \rangle \approx \langle g \rangle [h].$$

Unfortunately this is false. It's easy to show (using KR5, KR7 and standard modal reasoning) that the left to right implication holds, and indeed that it must hold for any path equation whatsoever. However the converse fails: we can arrive at a counterexample by considering the following simple model. The underlying multiframe of the model consists of three distinct points n, n_1 and n_2, such that $nR_f n_1$ and $nR_g n_2$. (Thus we have a simple 'V' shaped multiframe with n at the vertex.) Decorate nodes n_1 and n_2 with exactly the same atomic information: as they are both 'dead ends' this means they make precisely the same formulas true. To put matters another way, it means that both n_1 and n_2, although distinct, are associated with the same MCS, namely the set of formulas both make true. Call this MCS h'. Next consider node n. As n_1 and n_2 are distinct points that n is appropriately related to, we have that $\neg(\langle f \rangle \approx \langle g \rangle)$ is true at n. Further, for all ϕ such that $\phi \in h'$, our little model verifies both $\langle f \rangle \phi$ and $\langle g \rangle \phi$ at n. Let h be the MCS consisting of all the wffs true in our little model at n. But both h' and h are nodes in the Henkin model $\mathbf{M}^H$, and when we think about what's happening in the Henkin model we have our counterexample. By construction we have that in the Henkin model $hR_f h'$ and $hR_g h'$, thus we have that $\mathbf{M}^H \models \langle f \rangle \approx \langle g \rangle [h]$. But $\neg(\langle f \rangle \approx \langle g \rangle) \in h$, and as h is an MCS this means $\langle f \rangle \approx \langle g \rangle \notin h$.

Thus the straightforward Henkin method won't work, so we'll need to be more delicate. In the following pages I'll show in some detail how a method used by Burgess [6] for various tense logics can be adapted to fit L^{KR}. The method is an inductive process which builds something called a *chronicle*. As we shall see, given a chronicle with certain properties a suitable L^{KR}

model lies to hand. It's worth remarking that we're not forced into this chronicle construction method: Moss [17] gives an alternative 'Segerberg style' construction in which the faulty Henkin frame is massaged into shape.

The following conventions will prove extremely useful. Suppose we are working with some multiframe $\mathbf{N}$ or some model $\mathbf{M}$ on $\mathbf{N}$. Then if P is a path in $\mathbf{N}$, by $\langle P \rangle$ is meant the sequence of modalities that corresponds to P. For example, if P is the path $nR_f n_1 R_g n'$, then $\langle P \rangle$ is $\langle f \rangle \langle g \rangle$. On the other hand, given a sequence of modalities $\langle F \rangle$ and some $n \in N$, by F is meant the path in $\mathbf{N}$ starting at n, if this path exists. We only use this latter convention when we know the relevant path exists.

Definition 4.1 *If* $\mathbf{N}$ *is a multiframe, a* chronicle C *on* $\mathbf{N}$ *is a function* C *from* N *into the Henkin multiframe.* C *is* coherent *on* $\mathbf{N}$ *iff* C *is an order preserving morphism into the Henkin multiframe.* C *is* perfect *on* $\mathbf{N}$ *iff* $\langle f \rangle \phi \in C(n)$ *implies that there is an* $n' \in N$ *such that* $nR_f n'$ *and* $\phi \in C(n')$. C *is* attesting *on* $\mathbf{N}$ *iff for all* $n_1, n_2 \in N$, *and all paths* P, Q *in* N, $n_1 P n_2$ *and* $n_1 Q n_2$ *implies* $\langle P \rangle \approx \langle Q \rangle \in C(n_1)$. C *is* fulfilled *on* $\mathbf{N}$ *iff for all* n_1, n_2 $n_3 \in N$, *if* $\langle F \rangle \approx \langle G \rangle \in C(n_1)$, F *and* G *exist in* N, $n_1 F n_2$ *and* $n_1 G n_3$, *then* $n_2 = n_3$. □

The terminology of chronicles, coherent chronicles and perfect chronicles is Burgess's. Beyond noting that a chronicle C on $\mathbf{N}$ is coherent iff for all n, $n' \in N$, for all wffs ϕ, $nR_f n'$ and $\phi \in C(n')$ implies $\langle f \rangle \phi \in C(n)$, I won't discuss this machinery here — the ideas underlying it are standard in modal logic. Attesting chronicles and fulfilled chronicles are new. In a sense they are analogs of coherency and perfection for dealing with path equations: their role will become clearer as we proceed.

Definition 4.2 *If* C *is a chronicle on a multiframe* $\mathbf{N}$, *the* L^{KR} *model* induced *by* C *is* $\langle \mathbf{N}, \{Q_\alpha\}_{\alpha \in \mathcal{A}} \rangle$, *where* $Q_\alpha = \{n : p_\alpha \in C(n)\}$. □

Lemma 4.1 (Truth Lemma) *Let* C *be a coherent, perfect, attesting and fulfilled chronicle on a multiframe* $\mathbf{N}$. *Let* $\mathbf{M}_C$ *be the model induced by* C. *Then* $\mathbf{M}_C \models \phi[n]$ *iff* $\phi \in C(n)$.

Proof:

By induction on the structure of ϕ. The case involving propositional variables, and those involving the Booleans are trivial. The fact that C is coherent and perfect drives through the modal application case. The fact that C is coherent, attesting and fulfilled, together with the fact (which

follows from KR5) that $\langle F\rangle \approx \langle G\rangle \in C(n)$ implies that both $\langle F\rangle\top$ and $\langle G\rangle\top$ are in $C(N)$, handles the path equation case. □

Thus model building has been reduced to chronicle construction. In order to prove the completeness theorem we can proceed by building a chronicle with the four desirable properties listed in the previous lemma, and then use the induced model. This is what we'll now do, using an inductive construction.

The bric-a-brac underlying the construction is as follows. We fix a denumerably infinite set $\mathcal{N}$. Let Λ be the set of all pairs $\langle \mathbf{N}, C\rangle$ such that $\mathbf{N}$ is a partial functional multiframe $\langle N, \{R_l\}_{l\in\mathcal{L}}\rangle$, $N \subseteq \mathcal{N}$, and C is a coherent chronicle on $\mathbf{N}$. Given two elements $\lambda = \langle \mathbf{N}, C\rangle$ and $\lambda' = \langle \mathbf{N}', C'\rangle$ of Λ, we say λ' *extends* λ iff $\mathbf{N}$ is a subframe of $\mathbf{N}'$ and $C \subseteq C'$. Let ϕ be any wff and $n \in \mathcal{N}$. Then the pair $\langle n, \langle f\rangle\phi\rangle$ is called a *requirement*. Given any $\lambda = \langle \mathbf{N}, C\rangle$, we say the requirement $\langle n, \langle f\rangle\phi\rangle$ is *alive* for λ iff n is a node in $\mathbf{N}$, $\langle f\rangle\phi \in C(n)$, and there is no node n' of $\mathbf{N}$ such that nR_fn' and $\phi \in C(n')$. We say that the requirement $\langle n, \langle f\rangle\phi\rangle$ is *dead* for λ iff n is a node in $\mathbf{N}$, $\langle f\rangle\phi \in C(n)$ and there is some node n' of $\mathbf{N}$ such that nR_fn' and $\phi \in C(n')$. Note that if a requirement is dead for λ then it is dead in any extension of λ; death is monotonically increasing.

We are now ready for the lemma that lies at the heart of this method. Our ultimate goal is to build a chronicle having the four properties listed in the truth lemma, and we're going to use the elements of Λ for this purpose. Typically, however, the chronicles present in members of Λ won't be perfect. To put it another way, there'll typically be requirements alive for elements λ of Λ. The next lemma shows how to 'kill off' these requirements — thus coming closer to achieving a perfect chronicle — while retaining the pleasant properties already achieved:

Lemma 4.2 (Killing Lemma) *Let $\lambda = \langle N, \{R_l\}_{l\in\mathcal{L}}, C\rangle$ be an element of Λ such that $\mathbf{N}$ is generated by some point $n_0 \in N$, and C is both attesting and fulfilled on $\mathbf{N}$. Then for any requirement $\langle n, \psi\rangle$ which is alive for λ there exists a triple $\lambda' = \langle N', \{R'_l\}_{l\in\mathcal{L}}, C'\rangle$ in Λ such that λ' is an extension of λ, $\mathbf{N}'$ is generated by n_0, C' is both attesting and fulfilled on $\mathbf{N}'$, and the requirement $\langle n, \psi\rangle$ is dead for λ'.*

Proof:

Suppose the requirement is $\langle n, \langle f\rangle\phi\rangle$. Let X be any path from n_0 to n; at least one must exist as $\mathbf{N}$ is generated by n_0. As C is coherent on

$\mathbf{N}$ $\langle X\rangle\langle f\rangle\phi \in C(n_0)$. Now there are two possibilities. Either for some path X such that n_0Xn there exists a path W starting at n_0 such that $\langle W\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$ or there are no such pairs of paths X and W. We consider each case separately.

Case 1. Assume there are no such pairs of paths X and W in $\mathbf{N}$. Let $n' \in \mathcal{N}\backslash N$ and $\Sigma = \{\theta : [f]\theta \in C(n)\}$. Define:

$$\begin{array}{lcl} N' & = & N \cup \{n'\} \\ R'_f & = & R_f \cup \{\langle n, n'\rangle\} \\ R'_g & = & R_g, \text{ for all } g \in \mathcal{L} \text{ such that } g \neq f. \\ C' & = & C \cup \{\langle n, \Sigma^\infty\rangle\} \end{array}$$

Let $\lambda' = \langle N', \{R'_l\}_{l\in\mathcal{L}}, C'\rangle$.

Trivially for all $g \in \mathcal{L}$ such that $g \neq f$, R'_g is partial functional. Moreover R'_f must be partial functional as well, for if it were not then for some $m \in N$, nR_fm. But as C is coherent on $\mathbf{N}$ this is impossible, for it would mean that the requirement $\langle n, \langle f\rangle\phi\rangle$ was already dead in λ, contradicting our original assumption. Moreover by our choice of Σ it follows by standard modal logical arguments that C' is coherent on $\mathbf{N}'$. Thus $\lambda' \in \Lambda$ and indeed λ' is an extension of λ. It is clear that $\mathbf{N}'$ is generated by n_0, and by construction the requirement $\langle n, \langle f\rangle\phi\rangle$ is dead in λ'. Thus it only remains to show that C' is both attesting and fulfilled on $\mathbf{N}'$.

Suppose C' is not attesting on $\mathbf{N}'$. Then there are points n_1, $n_2 \in N'$ and paths P and Q in $\mathbf{N}'$ such that n_1Pn_2 and n_1Qn_2 and $\langle P\rangle \approx \langle Q\rangle \notin C'(n_1)$. Now P and Q cannot be paths in $\mathbf{N}$ as C is attesting on $\mathbf{N}$, therefore at least one of the two paths must contain n'. But by construction a path in $\mathbf{N}'$ can contain n' iff it terminates at n', hence as both P and Q terminate at the same place, both of them terminate at n' and n_2 must be n'. Next note that n_1 cannot equal n' for $0 \approx 0$ is an axiom and hence is in $C(n_1)$, so this degenerate case cannot have been where attestation failed. This means that $n_1 \in N$ and thus both P and Q are non-null paths. So we can write P as $P'R'_f$, where $n_1P'n$ and nR'_fn', and Q as $Q'R'_f$ where $n_1Q'n$ and nR'_fn'. Note that both P' and Q' are paths in $\mathbf{N}$. Not both of them can be null, for as $\langle f\rangle\phi \in C(n)$ then by KR4 $\langle f\rangle \approx \langle f\rangle \in C(n)$, but this wff is just $\langle P\rangle \approx \langle Q\rangle$ given that both P' and Q' are null, and we have contradicted our assumption of attestation failure. So suppose without loss of generality that P' is a non-null path. Now as C is attesting on $\mathbf{N}$, then as $n_1P'n$ and $n_1Q'n$ then $\langle P'\rangle \approx \langle Q'\rangle \in C(n_1)$. As C is coherent $\langle P'\rangle\langle f\rangle\phi \in C(n_1)$. So by KR8, $\langle P'\rangle\langle f\rangle \approx \langle Q'\rangle\langle f\rangle \in C(n_1)$. That is, $\langle P\rangle \approx \langle Q\rangle \in C(n_1)$, contradicting our assumption. Hence C' is attesting on $\mathbf{N}'$.

Suppose C' is not fulfilled on $\mathbf{N}'$. Then there is an $n_1 \in N$ such that $\langle F\rangle \approx \langle G\rangle \in C(n_1)$, F and G exist in $\mathbf{N}$, $n_1 F n_2$ and $n_1 G n_3$ and $n_2 \neq n_3$. First we observe that this is impossible if $n_1 = n'$, hence $n_1 \in N$. Second, observe that since C is fulfilled on $\mathbf{N}$ at least one of F or G contains n'. However a path in $\mathbf{N}'$ contains n' iff it terminates there. As by assumption F and G terminate at different nodes, exactly one of them must terminate at n'. Suppose without loss of generality that it is G that terminates at n', that is $n_3 = n'$. Then we can express G as $G'R'_f$ where $n_1 G' n$ and $nR'_f n'$, and G' is a path in $\mathbf{N}$. Now as F does not terminate at n' it is a path in $\mathbf{N}$. By assumption $\langle F\rangle \approx \langle G\rangle \in C(n_1)$. That is $\langle F\rangle \approx \langle G'\rangle\langle f\rangle \in C(n_1)$. Let X be a path in $\mathbf{N}$ from n_0 to n_1. As C is coherent $\langle X\rangle(\langle F\rangle \approx \langle G'\rangle\langle f\rangle) \in C(n_0)$. So by KR6 $\langle X\rangle\langle F\rangle \approx \langle X\rangle\langle G'\rangle\langle f\rangle \in C(n_1)$. But as XG' is a path in N from n_0 to n we have contradicted the assumption underlying case 1 and thus C' must be fulfilled on $\mathbf{N}'$.

Case 2. Suppose that for at least one path X such that $n_0 X n$ there is a path W in $\mathbf{N}$ starting at n_0 such that $\langle W\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$. Fix the choice of some such X and let m be the endpoint of path W. First we observe that if W' is any other path in $\mathbf{N}$ such that $\langle W'\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$ and $n_0 W' m'$ then $m = m'$. For it follows from KR2 and KR3 that $\langle W\rangle \approx \langle W'\rangle \in C(n_0)$, thus as C is fulfilled on $\mathbf{N}$, $m = m'$. Define:

$$\begin{array}{lcl} N' & = & N \\ R'_f & = & R_f \cup \{\langle n, m\rangle\} \\ R'_g & = & R_g, \text{ for all } g \in \mathcal{L} \text{ such that } g \neq f \\ C' & = & C \end{array}$$

Let λ' be $\langle N', \{R_l\}_{l\in\mathcal{L}}, C'\rangle$.

It is clear that for all $g \in \mathcal{L}$ R'_g is partial functional. Moreover C' is coherent on $\mathbf{N}'$. To show this we need merely show that for all $\psi \in C(m)$, $\langle f\rangle\psi \in C(n)$, as by hypothesis C is coherent on $\mathbf{N}$. So let $\psi \in C(m)$. By the coherency of C on $\mathbf{N}$, $\langle W\rangle\psi \in C(n_0)$. By assumption $\langle W\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$, so by KR7 $\langle X\rangle\langle f\rangle\psi \in C(n_0)$. Because all instances of $\langle X\rangle\phi \rightarrow [X]\phi$ are provable, this means $[X]\langle f\rangle\psi \in C(n_0)$. Now if $\langle f\rangle\psi \notin C(n)$ then by the coherency of C on $\mathbf{N}$, $\langle X\rangle\neg\langle f\rangle\psi \in C(n_0)$. That is, $\neg[X]\langle f\rangle\psi \in C(n_0)$ and we have that $C(n_0)$ is inconsistent which is not possible. Thus $\langle f\rangle\psi \in C(n)$ and C' is coherent on $\mathbf{N}'$.

So $\lambda' \in \Lambda$, and in fact λ' is an extension of λ. Moreover it is clear by construction that $\mathbf{N}'$ is generated by n_0 and the requirement $\langle n, \langle f\rangle\phi\rangle$ is dead in λ'. Thus it only remains to show that C' is attesting and fulfilled on

$\mathbf{N}'$. In order to make further progress, the following sublemma is required:

> **Detour Lemma:** *If D is a path in $\mathbf{N}'$ such that nDm, then $\langle D\rangle \approx \langle f\rangle \in C(n)$.*

We will now prove this sublemma by induction on the number of links nR_fm in D. Note that the inductive step must start on the assumption that there are at least two such links in D, thus two base cases must be established.

So first suppose that D contains no such links, that is, D is a path in $\mathbf{N}$. As XD and W are both paths in $\mathbf{N}$ from n_0 to m then as C is attesting on $\mathbf{N}$, $\langle W\rangle \approx \langle X\rangle\langle D\rangle \in C(n_0)$. By assumption $\langle W\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$, so by KR2 and KR3 we have that $\langle X\rangle\langle D\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$. So by KR6 $\langle X\rangle(\langle D\rangle \approx \langle f\rangle) \in C(n_0)$ and thus by the coherency of C on $\mathbf{N}$ $\langle D\rangle \approx \langle f\rangle \in C(n)$.

Next suppose that D contains precisely one link nR'_fm. Then we can express D as SR'_fT where nSn, mTm and both S and T are paths in N. We need to show that $\langle S\rangle\langle f\rangle\langle T\rangle \approx \langle f\rangle \in C(n)$. Now as W and WT are both paths in $\mathbf{N}$ from n_0 to m, then as C is attesting on $\mathbf{N}$ $\langle W\rangle\langle T\rangle \approx \langle W\rangle \in C(n_0)$. By the case 2 assumption $\langle W\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$, so by two uses of KR9 (together with uses of KR2 and KR3), we have that $\langle X\rangle\langle f\rangle\langle T\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$. But as X and XS are both paths in $\mathbf{N}$ leading from n_0 to n, $\langle X\rangle\langle S\rangle \approx \langle X\rangle \in C(n_0)$ as C is attesting on $\mathbf{N}$. So by another use of KR9 we have that $\langle X\rangle\langle S\rangle\langle f\rangle\langle T\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$. It then follows using KR6 and the coherency of C on $\mathbf{N}$ that $\langle S\rangle\langle f\rangle\langle T\rangle \approx \langle f\rangle \in C(n)$ as required.

So suppose the result holds for all paths D' from n to m such that D' contains less than r occurrences of nR'_fm, where $2 \leq r$. Let D be a path from n to m containing exactly r occurrences of nR'_fm. First we express D as SBT where nSm, mBn and nTm where S and T each contain at least one occurrence of the link nR'_fm and B contains no such occurrences. Note that both S and T satisfy the inductive hypothesis (hereafter IH) and that B is a path in $\mathbf{N}$.

By IH, $\langle T\rangle \approx \langle f\rangle \in C(n)$. Hence using KR6 and the coherency of C on $\mathbf{N}$, $\langle X\rangle\langle T\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$. As WB and X are both paths in $\mathbf{N}$ from n_0 to n, then as C is attesting on $\mathbf{N}$, $\langle W\rangle\langle B\rangle \approx \langle X\rangle \in C(n_0)$. So using KR9 we have that $\langle W\rangle\langle B\rangle\langle T\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$; call this statement $*$. It also holds by IH that $\langle S\rangle \approx \langle f\rangle \in C(n)$, hence using the coherency of C on $\mathbf{N}$ together with KR6 we have that $\langle X\rangle\langle S\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$. But the case 2 assumption is that $\langle W\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$, so using KR2 and KR3 we have

that $\langle X\rangle\langle S\rangle \approx \langle W\rangle \in C(n_0)$. But using this fact together with $*$ just above and KR9 yields $\langle X\rangle\langle S\rangle\langle B\rangle\langle T\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$. It then follows using KR6 and the coherency of C on $\mathbf{N}$ that $\langle S\rangle\langle B\rangle\langle T\rangle \approx \langle f\rangle \in C(n)$ which is what we require. This establishes the inductive step. Thus the truth of the Detour Lemma follows by induction on r.

With the help of this sublemma it is relatively straightforward (though tedious) to establish by a reductio-ad-absurdum argument that C' is both attesting and fulfilled on $\mathbf{N}'$, and we now turn to these tasks. Both tasks split naturally into two subtasks.

C' is attesting on $\mathbf{N}'$. For suppose it is not. Then there are n_1, $n_2 \in N'$ and paths Q and Q' in $\mathbf{N}'$ such that $n_1 Q n_2$ and $n_1 Q' n_2$ and $\langle Q\rangle \approx \langle Q'\rangle \notin C(n_1)$. Now as C is attesting on $\mathbf{N}$ this is only possible if at least one of Q and Q' contains an occurrence of $nR'_f m$.

First suppose that both Q and Q' contain such a link. Express Q as SDT such that $n_1 Sn$, nDm and mTn_2 where both S and T are paths in $\mathbf{N}$. In analogous fashion express Q' as $S'D'T'$. Now as T and T' are both paths in $\mathbf{N}$, and as C is attesting on $\mathbf{N}$, $\langle T\rangle \approx \langle T'\rangle \in C(m)$. By the coherency of C' on $\mathbf{N}'$, $\langle f\rangle(\langle T\rangle \approx \langle T'\rangle) \in C(n)$. But by the detour lemma we have that $\langle D\rangle \approx \langle f\rangle \in C(n)$, so by KR7 we have that $\langle D\rangle(\langle T\rangle \approx \langle T'\rangle) \in C(n)$, and thus by KR6 $\langle D\rangle\langle T\rangle \approx \langle D\rangle\langle T'\rangle \in C(n)$. By another application of the Detour Lemma we have that $\langle D'\rangle \approx \langle f\rangle \in C(n)$; as $\langle D\rangle \approx \langle f\rangle \in C(n)$ then using KR3 we have that $\langle D\rangle \approx \langle D'\rangle \in C(n)$. Thus using KR9 yields $\langle D\rangle\langle T\rangle \approx \langle D'\rangle\langle T'\rangle \in C(n)$. Now as C is coherent on $\mathbf{N}$, $\langle S\rangle(\langle D\rangle\langle T\rangle \approx \langle D\rangle'\langle T\rangle') \in C(n_1)$, so by KR6 $\langle S\rangle\langle D\rangle\langle T\rangle \approx \langle S\rangle\langle D'\rangle\langle T'\rangle \in C(n_1)$. But as S and S' are both paths in $\mathbf{N}$ from n_0 to n then as C is attesting on $\mathbf{N}$ we have that $\langle S\rangle \approx \langle S'\rangle \in C(n)$, so by KR9 we have that $\langle S\rangle\langle D\rangle\langle T\rangle \approx \langle S'\rangle\langle D'\rangle\langle T'\rangle \in C(n)$. That is, $\langle Q\rangle \approx \langle Q'\rangle \in C(n)$ and we have a contradiction.

So the only possibility remaining if attestation is to fail is that exactly one of Q or Q' contains a link $nR'_f m$. Suppose without loss of generality that only Q contains such a link. Express Q as SDT where $n_1 Sn$, nDm and mTn_2 and both S and T are paths in $\mathbf{N}$. As WT is a path in $\mathbf{N}$ from n_0 to n_2 then as C is attesting on $\mathbf{N}$ $\langle W\rangle\langle T\rangle \approx \langle W\rangle\langle T\rangle \in C(n_0)$. But then as the case 2 assumption is that $\langle W\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$ we have, using KR9, that $\langle W\rangle\langle T\rangle \approx \langle X\rangle\langle f\rangle\langle T\rangle \in C(n_0)$. Now by the Detour Lemma $\langle f\rangle \approx \langle D\rangle \in C(n)$, thus $\langle X\rangle\langle f\rangle \approx \langle X\rangle\langle D\rangle \in C(n_0)$ by KR6 and the coherency of C on $\mathbf{N}$. So again using KR9 we have that $\langle W\rangle\langle T\rangle \approx \langle X\rangle\langle D\rangle\langle T\rangle \in C(n_0)$. Now let P be any path in $\mathbf{N}$ from n_0 to n_1. Then as X and PS are both paths

in $\mathbf{N}$ from n_0 to n than as C is attesting on $\mathbf{N}$, $\langle X\rangle \approx \langle P\rangle\langle S\rangle \in C(n_0)$. So again by KR9 $\langle W\rangle\langle T\rangle \approx \langle P\rangle\langle S\rangle\langle D\rangle\langle T\rangle \in C(n_0)$. Now WT and XQ' are both paths in $\mathbf{N}$ from n_0 to n_2, so again using the fact that C is attesting on $\mathbf{N}$, $\langle W\rangle\langle T\rangle \approx \langle P\rangle\langle Q'\rangle \in C(n_0)$. Hence $\langle P\rangle\langle Q'\rangle \approx \langle P\rangle\langle S\rangle\langle D\rangle\langle T\rangle \in C(n_0)$. Thus by KR6 and the coherency of of C on $\mathbf{N}$, $\langle Q'\rangle \approx \langle S\rangle\langle D\rangle\langle T\rangle \in C(n_1)$, that is, $\langle Q'\rangle \approx \langle Q\rangle \in C(n_1)$ and we have another contradiction. Thus there is no possible way that attestation can fail and so C' must be attesting on $\mathbf{N}'$.

Finally, C' is fulfilled on $\mathbf{N}'$. For suppose it is not. That is, suppose there are n_1, n_2 and $n_3 \in N'$ such that $\langle F\rangle \approx \langle G\rangle \in C(n_1)$, F and G exist in $\mathbf{N}'$, $n_1 F n_2$, $n_1 G n_3$ and $n_2 \neq n_3$. As C is fulfilled on $\mathbf{N}$ this can only happen if at least one of F or G contains a link $nR'_f m$.

First suppose that both F and G contain such a link. Express F as SDT where $n_1 S n$, nDm, mTn_2 and both S and T are paths in $\mathbf{N}$. In similar fashion express G as $S'D'T'$. Now as WT and WT' are paths in $\mathbf{N}$ leading from n_0 to n_2 and n_3 respectively, then as $n_2 \neq n_3$ and C is fulfilled on $\mathbf{N}$ we have $\neg(\langle W\rangle\langle T\rangle \approx \langle W\rangle\langle T'\rangle) \in C(n_0)$; call this statement $*$. By the Detour Lemma $\langle f\rangle \approx \langle D\rangle$ and $\langle f\rangle \approx \langle D'\rangle$ are both in $C(n)$, so by KR6 and the coherency of C on $\mathbf{N}$, both $\langle X\rangle\langle f\rangle \approx \langle X\rangle\langle D\rangle$ and $\langle X\rangle\langle f\rangle \approx \langle X\rangle\langle D'\rangle \in C(n_0)$. By assumption $\langle W\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$, so $\langle W\rangle \approx \langle X\rangle\langle D\rangle$ and $\langle W\rangle \approx \langle X\rangle\langle D'\rangle$ are both in $C(n_0)$. But then using KR9 and $*$ twice we have $\neg(\langle X\rangle\langle D\rangle\langle T\rangle \approx \langle X\rangle\langle D'\rangle\langle T'\rangle) \in C(n_0)$. Now let P be any path in $\mathbf{N}$ from n_0 to n_1. As X, PS and PS' are all paths in $\mathbf{N}$ from n_0 to n then as C is attesting on $\mathbf{N}$, $\langle X\rangle \approx \langle P\rangle\langle S\rangle$ and $\langle X\rangle \approx \langle P\rangle\langle S'\rangle$ are both in $C(n_0)$. Again using KR9 we have that $\neg(\langle P\rangle\langle S\rangle\langle D\rangle\langle T\rangle \approx \langle P\rangle\langle S'\rangle\langle D'\rangle\langle T'\rangle) \in C(n_0))$. But this means that $\neg(\langle S\rangle\langle D\rangle\langle T\rangle \approx \langle S'\rangle\langle D'\rangle\langle T'\rangle) \in C(n_1)$. That is, $\neg(\langle F\rangle \approx \langle G\rangle) \in C(n_1)$ and we have a contradiction.

So only one of F or G can contain the link $nR'_f m$. Suppose without loss of generality that it is F that contains it. As in the previous case express F as SDT. Let P be any path in $\mathbf{N}$ from n_0 to n_1. As WT and PG are both paths in $\mathbf{N}$ leading from n_0 to n_2 and n_3 respectively, then as $n_2 \neq n_3$ and C is fulfilled on $\mathbf{N}$ we have that $\neg(\langle W\rangle\langle T\rangle \approx \langle P\rangle\langle G\rangle) \in C(n_0)$; call this statement $*$. The case 2 assumption is that $\langle W\rangle \approx \langle X\rangle\langle f\rangle \in C(n_0)$. Using the Detour Lemma, the coherency of C on $\mathbf{N}$ and KR6 we have that $\langle X\rangle\langle f\rangle \approx \langle X\rangle\langle D\rangle \in C(n_0)$. Hence $\langle W\rangle \approx \langle X\rangle\langle D\rangle \in C(n_0)$. Using this fact together with KR9 and $*$ we have that $\neg(\langle X\rangle\langle D\rangle\langle T\rangle \approx \langle P\rangle\langle G\rangle) \in C(n_0)$. By the attestation of C on $\mathbf{N}$ $\langle X\rangle \approx \langle P\rangle\langle S\rangle \in C(n_0)$. Therefore $\neg(\langle P\rangle\langle S\rangle\langle D\rangle\langle T\rangle \approx \langle P\rangle\langle G\rangle) \in C(n_0)$. Therefore $\neg\langle S\rangle\langle D\rangle\langle T\rangle \approx \langle S\rangle \in C(n_1)$. In short, $\neg(\langle F\rangle \approx \langle G\rangle) \in C(n_1)$ and we have another contradiction.

Thus C' is fulfilled on $\mathbf{N}'$.

So we have established all we set out to do and the Killing Lemma is proved. □

Establishing this result required some effort. It's worth remarking that from the point of view of tableaux theorem proving, the effort was not expended in vain: the (constructively flavoured) details of the proof are quite revealing. In particular, deciding whether a Case 1 type extension or a Case 2 type extension is called for is one of the fundamental control decisions that would have to be built into such systems. However, axiomatisations are our present concern, and from an axiomatic perspective what the Killing Lemma leads to is:

Theorem 4.1 (Completeness Theorem) *Every consistent set of sentences has a model.*

Proof:

Suppose we have fixed $\mathcal{N}$ and defined Λ as discussed earlier. Enumerate the requirements so that in the sequel it makes sense to talk of 'the least requirement alive for λ'. Given a consistent set of sentences Σ, form Σ^∞. Choose an arbitrary $n_0 \in \mathcal{N}$. Define λ^0 to be $\langle N^0, \{R_l^0\}_{l\in\mathcal{L}}, C^0\rangle$, where $N^0 = \{n_0\}$; for all $l \in \mathcal{L}$ $R_l^0 = \{\langle n_0, n_0\rangle\}$ iff $\langle l\rangle \approx 0 \in \Sigma^\infty$, and equals $\emptyset$ otherwise; and $C^0 = \{\langle n_0, \Sigma^\infty\rangle\}$. It is easy to see that $\lambda^0 \in \Lambda$ and that C is attesting and fulfilled on $\mathbf{N}^0$.

Suppose λ^k has been defined. Either some requirement is alive for λ^k, or all requirements are dead for λ^k. In the first case define λ^{k+1} to be the result of killing off the least such live requirement as shown in the Killing Lemma, otherwise define λ^{k+1} to be λ^k. Finally, define λ^+ to be $\langle N^+, \{R_l^+\}_{l\in\mathcal{L}}, C^+\rangle$ where $N^+ = \bigcup_{k\in\omega} N^k$; for all $l \in \mathcal{L}$ $R_l^+ = \bigcup_{k\in\omega} R_l^k$; and $C^+ = \bigcup_{k\in\omega} C^k$. $\mathbf{N}^+$ is a partial functional multiframe, for if it were not then for some $k \in \omega$ $\mathbf{N}^k$ would not be partial functional, which is impossible. Clearly C^+ is a chronicle on $\mathbf{N}^+$. By construction it is perfect. It is also coherent, attesting and fulfilled — for if it were not we would have lost these properties at some finite stage, which is impossible. Hence C^+ satisfies the conditions of the Truth Lemma and thus the model it induces on $\mathbf{N}^+$ verifies every sentence in Σ at n_0. □

Extending this completeness result to the other classes of AV structures used in computational linguistics is very easy. For example, to axiomatise

those AVSs satisfying the constant-constant clash, we add as axioms all L^{KR} instances of *Ccc*, and to axiomatise those AVSs satisfying the constant-compound clash we add all L^{KR} instances of *Term*. To insist that each particular piece of atomic information is instantiated at a unique node we add all instances of $\langle F\rangle p_\alpha \wedge \langle G\rangle p_\alpha \rightarrow \langle F\rangle \approx \langle G\rangle$. Less trivially, to axiomatise the class of acyclic AV structures we add all instances of $\neg(0 \approx \langle F\rangle)$. Proving that these various additions suffice merely involves making extra checks in the proof of the Killing Lemma. These results are additive, so we have the usual menu of logical options concerning AV structures at our disposal in L^{KR}.

Finally, it should be clear that the method of selective generation proves that the basic logic, and all the extensions just mentioned, have the finite model property. It then follows by the usual arguments that these logics are all decidable.

5 Concluding remarks

In L, L^{KR} and L^N we have a group of modal languages which mirror the most common Attribute Value formalisms devised by computational linguists. Given the particularly direct nature of the correspondences involved, it seems reasonable to claim that propositional modal languages are the working languages of much computational linguistics.

It seems possible to build on these correspondences. First, there are other Attribute Value formalisms in the literature which seem amenable to modal analysis. Second, with the link between attribute value formalisms and modal languages established, it becomes possible to investigate ideas from modal logic with a view to applying them in computational linguistics, and in fact there are several modally natural extensions of L, L^{KR} and L^N (for example, adding a universal modality) that seem to be linguistically useful. Third, propositional dynamic logic — modal logic's big brother — beckons, and indeed those computational linguists who use *functional uncertainty* have already heeded the call [15]. However an adequate discussion of these topics is not possible here, and so I'll close the paper on a more general note.

This investigation has shown that modal languages can arise quite naturally when one tries to formulate simple constraint languages for talking about information systems. It is easy to see why. Many types of information systems are fairly simple graphical structures, and modal languages are the simplest languages for talking about graphs. Given that this is so, it seems

natural to investigate whether existing constraint languages from other domains can be usefully examined from the modal perspective. Intriguingly, at least two common knowledge representation formalisms — the *frames* of Artificial Intelligence, and the *isa* hierarchies of cognitive psychology — seem open to a modal treatment. Only further work can establish whether this is a good way of looking at these formalisms, but the idea seems a natural one and may be worth pursuing.

Acknowledgements I would like to thank Bob Carpenter, Claire Gardent, Gerald Gazdar, Valentin Goranko, Bob Kasper, Marcus Kracht, Larry Moss, Carl Pollard, Maarten de Rijke, Jerry Seligman, Edith Spaan and the referee for their comments on this paper and much useful discussion besides. I am also grateful for the hospitality of the Faculteit der Wiskunde en Informatica, Universiteit van Amsterdam, where this paper was written, and for the financial support of the SERC, the Science and Engineering Research Council of the United Kingdom.

References

[1] S. Bird and P. Blackburn, 1991, A Logical Approach to Arabic Phonology, in *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics*, Berlin.

[2] P. Blackburn, 1989, Nominal Tense Logic, ITLI Prepublication Series LP-90-05, Amsterdam.

[3] P. Blackburn, 1990, *Nominal Tense Logic and other Sorted Intensional Frameworks*, PhD Thesis, Centre for Cognitive Science, University of Edinburgh, Scotland.

[4] P. Blackburn and E. Spaan, 1991, On the Complexity of Attribute Value Logics, to appear in *Proceedings of the Eighth Amsterdam Colloquium.*

[5] R. Bull, 1968, An Approach to Tense Logic, *Theoria*, **12**, pages 171–182.

[6] J. Burgess, 1984, Basic Tense Logic, in *Handbook of Philosophical Logic*, volume 2, edited by D. Gabbay and F. Guenthner, Reidel.

[7] G. Gargov, S. Passy and T. Tinchev, 1987, Modal Environment for Boolean Speculations, in *Mathematical Logic and its Applications*,

edited by D. Skordev, Proceedings of the 1986 Godel Summer School and Conference, Bulgaria, Plenum Press.

[8] G. Gargov and V. Goranko, 1989, Modal Logic with Names I, To appear in *Journal of Philosophical Logic.*

[9] G. Gazdar and G. Pullum, 1987, A logic for Category Definition, Cognitive Science, Research Paper CSRP 072, University of Sussex, United Kingdom.

[10] G. Gazdar, G. Pullum, R. Carpenter, E. Klein, T. Hukari and R. Levine, 1988, Category Structures, *Computational Linguistics*, **14**, pages 1–19.

[11] G. Hughes and M. Cresswell, 1984, *A Companion to Modal Logic*, Methuen & Co. Ltd., London.

[12] M. Johnson, 1988, *Attribute-Value Logic and the Theory of Grammar*, CSLI Lecture Notes Series, University of Chicago Press.

[13] L. Karttunen, 1984, Features and Values, in *Proceedings of the 10th International Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics*, Stanford, California, pages 28–33.

[14] R. Kasper and W. Rounds, 1986, A logical semantics for feature structures, in *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, Columbia University, New York, pages 257–266.

[15] B. Keller, 1991, *Feature Logics, Infinitary Descriptions and The Logical Treatment of Grammar*, PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, United Kingdom.

[16] M. Kracht, 1989, On the Logic of Category Definition, *Computational Linguistics*, **15**, pages 111–113.

[17] L. Moss, 1991, Completeness Theorems for Logics of Feature Structures, Indiana University Logic Group Preprint No. IULG-91-2, to appear in *Proceedings of the MSRI Workshop on Logic From Computer Science*, edited by Yiannis N. Moschovakis, Springer Verlag.

[18] F. Pereira, and S. Shieber, 1984, The semantics of grammar formalisms seen as computer languages, in *Proceedings of the 10th International*

Conference on Computational Linguistics and the 22nd Annual Meeting of the Association for Computational Linguistics, Stanford, California, pages 123–129.

[19] C. Pollard, Forthcoming, Sorts in unification-based grammar and what they mean, to appear in *Unification in Linguistic Analysis*, edited by M. Pinkal and B. Gregor.

[20] A. Prior, 1967, *Past, Present and Future*, Oxford University Press.

[21] M. Reape, 1991, *An Introduction to the Semantics of Unification-Based Grammar Formalisms,* DYANA deliverable R3.2.A, Centre for Cognitive Science, University of Edinburgh, Scotland.

[22] W. Rounds and R. Kasper, 1986, A Complete Logical Calculus for Record Structures Representing Linguistic Information, in *Proceedings of the 15th Annual Symposium on Logic in Computer Science*, Cambridge, Massachussets.

[23] W. Rounds and R. Kasper, 1990, The Logic of Unification in Grammar, *Linguistics and Philosophy*, **13**, pages 33–58.

[24] P. Ruhrberg, 1991, *Semantic Considerations for Constraint Based Grammar Formalisms,* Thesis, University of Bielefeld, Germany.

Department of Philosophy
Utrecht University
Heidelberglaan 8
3584 CS Utrecht
The Netherlands