NAMING WORLDS IN MODAL AND TEMPORAL LOGIC

D. M. GABBAY AND G. MALOD

Online version – May 2000

Abstract. In this paper we suggest adding to predicate modal and temporal logic a locality predicate W which gives names to worlds (or time points). We also study an equal time predicate $D(x, y)$ which states that two time points are at the same distance from the root. We provide the systems studied with complete axiomatizations and illustrate the expressive power gained for modal logic by simulating other logics. The completeness proofs rely on the fairly intuitive notion of a configuration in order to use a proof technique similar to a Henkin completion mixed with a tableau construction. The main elements of the completeness proofs are given for each case, while purely technical results are grouped in the appendix.

1. Introduction

It is accepted practice in classical logic to consider a fixed equality predicate $x = y$. Equality is a fixed non-logical symbol having a fixed interpretation, namely identity. First-order logic often comes with the equality predicate and the accompanying set of axioms known as the equality axioms. Historically other fixed non-logical predicates have been introduced, such as fixed linear orderings. The expressive power of such languages has been extensively studied in model theory and on many occasions such fixed predicates give the expressive power of infinitary connectives. The reason we consider additional fixed connectives, especially equality, is their usefulness in applications, and their widespread and fundamental nature. It makes sense to have equality and its axioms available whenever we use classical logic.

This paper offers similar additional predicates for modal and temporal logic. We propose that whenever modal or temporal logics are used, these additional predicates with their axioms should be available, just as equality is in first-order logic.

To explain our proposal, we need to present the constant domain Kripke semantics for modal and temporal logic. A model has the form $(\mathcal{M}, \alpha)$, $\mathcal{M} = (T, R, \mathcal{D}, h)$ being a structure, where T is the set of possible worlds, R is the accessibility relation, $\alpha \in T$ is the actual world. $\mathcal{D}$ is a first-order structure containing the domain D and interpretations of the functions symbols and h is the interpretation of the predicate symbols. $\models$ denotes the satisfaction relation.

We are now ready to propose our additional fixed predicates and their properties.

Keywords. hybrid logic, modal logic, nominals, predicate logic, temporal logic.

1. Identify the domain D of modal or temporal predicate logic with the set T of possible worlds. [1]
2. Use a predicate $W(x)$ which holds in a world α exactly when $\alpha = x$ i.e. $\alpha \models W(x)$ iff $\alpha = x$.
3. We would like to structure the set of possible worlds to be tree-like. The tree has a root which is the actual world. This means that each point has a distance from the actual world. We offer the equal-time predicate $D(x, y)$ which says that the points x and y have the same distance from the root.

This paper studies the syntax and semantics of such modal and temporal logics, with $W(x)$ and $D(x, y)$, as well as their applications. Proofs of the completeness theorems are relatively complex, but the method is of value to other possibly more subtle and refined logics.

The predicates $W(x)$ and $D(x, y)$ are motivated by the following application. Consider a temporal logic where moments of time are discrete, the past is linear and the future is branching. Such a flow of time arises in any system where alternative actions are available to move us from one state to another. In this model $W(x)$ simply names the world one is in and $D(x, y)$ says that x and y are two alternative futures, but at the same time distance from the beginning time. We can thus compare the different states we can be in after a given period of time, depending on the alternatives taken on the way.

The idea of naming the worlds or time points goes back to A. Prior [10]. A first appearance of this idea with quantification can be found in [4]. This theme has then been developed by different authors. [9] is a classic article on this topic which introduces nominals in dynamic logic, and includes a study of quantification over such nominals. Both [1] and [3] study a logic enhanced with nominals, with and without quantification respectively and in different settings. The predicate $W(x)$ is introduced in [7].

2. Predicate modal and temporal logic

2.1. **Predicate modal logic.** We give here a brief definition of a predicate modal logic. We assume it contains only one modal operator $\Diamond$, but to introduce other operators we just need to duplicate the clauses for $\Diamond$ with the necessary modifications. Further details about predicate modal logic can be found for example in [8].

2.1.1. *Syntax.* Let $\mathcal{S}$ be a classical first-order signature. In other words, $\mathcal{S}$ defines *function* symbols of various arities (with 0-ary function symbols often called *constants*) and *predicate* symbols of various arities.
Let $x_1, x_2, \ldots$ be a countable collection of *variable* symbols. We then define the set of *terms* as being the smallest set including all variable symbols and such that for any terms $t_1, \ldots, t_n$ and any n-ary function symbol f it also includes $f(t_1, \ldots, t_n)$. The *closed terms* are the terms where variable symbols do not appear. The set of *atomic formulas* is the set of words of the form $\top$ (the formula which is always true) or $p(t_1, \ldots, t_n)$ for an n-ary

[1] Identification is possible in the absence of equality because there are no axioms forcing the set of possible worlds or the domain to be finite.

predicate symbol p and terms $t_1, \ldots, t_n$. The set of *formulas* is the smallest set which includes all atomic formulas and such that for all formulas A and B it also includes $\neg A$, $A \wedge B$, $\forall x A$ (where x is a variable symbol) and $\Diamond A$. We define the usual concepts of *free* and *bound* variables and *substitutions*; a *sentence* is a formula without any free variable. We will also use the common abbreviations $\bot$ (the formula which is always false), $\vee$, $\rightarrow$, $\leftrightarrow$, $\exists x$ and $\Box$.

2.1.2. *Semantics.* We define the notion of a modal structure and what it means for a formula to be true. The domain is constant, function interpretations and variable assignments are rigid, i.e. they do not depend on the world considered. A modal structure is a 4-tuple $\mathcal{M} = (T, R_\Diamond, \mathcal{D}, h)$:

- T is a set of *time points.*
- $R_\Diamond$ is a binary relation on T.
- $\mathcal{D}$ is a first-order structure for the language with just the function symbols of $\mathcal{S}$; it therefore has a domain D and interpretations $\bar{f} : D^n \rightarrow D$ for each function symbol f.
- h is a map such that for any predicate symbol p, and for any time point α, $h(p, \alpha)$ is a subset of D^n.

We now have the "meaning" of functions, constants and predicates in that structure, but we must still give values to variables before we can decide whether a formula is true or false at a given point. This is done via an assignment v which maps variables to values in D. This assignment can then be naturally extended to all terms t using the inductive definition: $v(f(t_1, \ldots, t_n)) = \bar{f}(v(t_1), \ldots, v(t_n))$, where $\bar{f}$ is the interpretation of f given by $\mathcal{D}$.

Let $\mathcal{M}, \alpha, v \models A$ mean that the formula A holds at point α in the structure $\mathcal{M}$ with the assignment v. If u and v are assignments, we write $u \overset{x}{\sim} v$ if $\forall y \neq x, u(y) = v(y)$. We now define the satisfaction of a formula by induction:

$$\begin{aligned}
&\mathcal{M}, \alpha, v \models \top \\
&\mathcal{M}, \alpha, v \models p(t_1, \ldots, t_n) \text{ iff } (v(t_1), \ldots, v(t_n)) \in h(p, \alpha) \\
&\mathcal{M}, \alpha, v \models \neg A \text{ iff } \mathcal{M}, \alpha, v \not\models A \\
&\mathcal{M}, \alpha, v \models A \wedge B \text{ iff } \mathcal{M}, \alpha, v \models A \text{ and } \mathcal{M}, \alpha, v \models B \\
&\mathcal{M}, \alpha, v \models \forall x A \text{ iff } \forall u \overset{x}{\sim} v,\ \mathcal{M}, \alpha, u \models A \\
&\mathcal{M}, \alpha, v \models \Diamond A \text{ iff } \exists \beta \in T \text{ such that } \alpha R_\Diamond \beta \text{ and } \mathcal{M}, \beta, v \models A
\end{aligned}$$

If A is a sentence, it is clear that the choice of an assignment is unimportant, so we can write $\mathcal{M}, \alpha \models A$. We say that $(\mathcal{M}, \alpha)$ is a model of a sentence A if $\mathcal{M}, \alpha \models A$.

2.1.3. *A proof system.* We use the traditional Hilbert presentation of a proof, i.e. we give axioms and rules, and a *proof* of a formula A from the empty set is a sequence $A_1, \ldots, A_n$ such that $A_n = A$ and each A_i is an axiom or is obtained from the previous A_j using a deduction rule. If we have a proof of A from $\emptyset$ we write $\vdash A$, and say that A is a *theorem* of our logic. Given a set of assumptions Φ we say that A can be deduced from Φ (and we write $\Phi \vdash A$) if there is a sequence as above where each A_i is either in Φ or

is a theorem or is obtained by modus ponens from two previous elements. A formula A is said to be *consistent* if we cannot prove its negation ($\nvdash \neg A$). The deduction rules are:

(MP): modus ponens, from $A \to B$ and A deduce B.
(UG): universal generalization, from $A(x)$ deduce $\forall x A(x)$.
(MG): modal generalization, from A deduce $\Box A$.[2]

Here are the basic axioms:

(1): any tautology of classical propositional logic with formulas of our language substituted for atoms.
(2): quantifier axioms of classical predicate logic:
  (2a): $\forall x(A \to B) \to (A \to \forall x B)$, where x is not free in A.
  (2b): $\forall x A(x) \to A[t/x]$, where x does not appear free in the scope of a quantifier binding a variable from t.
(3): $\Box(A \to B) \to (\Box A \to \Box B)$.
(Barcan): $\forall x \Box A(x) \to \Box \forall x A(x)$.

2.2. **Predicate temporal logic.** We have two modal operators, $\Diamond_F$ (F stands for "in the Future") and $\Diamond_P$ (P stands for "in the Past"), and use $<$ instead of $R_{\Diamond_F}$. The relation $R_{\Diamond_P}$ is the converse of $<$. Syntax is defined as in the above section, with of course the two modal operators, and we define the semantics with the following changes:

- the relation $<$ on T must be "linear past, branching future", without any cycles; $(T, <)$ is called the flow of time.
- $\mathcal{M}, \alpha, v \models \Diamond_F A$ iff there is $\beta \in T$ such that $\alpha < \beta$ and $\mathcal{M}, \beta, v \models A$.
- $\mathcal{M}, \alpha, v \models \Diamond_P A$ iff there is $\beta \in T$ such that $\beta < \alpha$ and $\mathcal{M}, \beta, v \models A$.

Rule MG and axiom 3 are also duplicated for the future and past operators. See [5] for a more detailed presentation of predicate temporal logic.

3. Discrete temporal logic

We begin by studying the simpler case where the operators are not transitive. We therefore call them $\Diamond_T$ (T stands for "Tomorrow") instead of $\Diamond_F$ and $\Diamond_Y$ (Y stands for "Yesterday") instead of $\Diamond_P$. In this section we give an axiomatization and show that it is complete. In following sections we will show how we can adapt these axioms and proof to other cases.

3.1. **An axiomatization.** Here are the adapted basic modal rules and axioms and the axioms for the locality and same-depth predicates. In the following, A and B are formulas, x is a variable, t is a term, m, n, p, q are positive natural numbers.
Let us first introduce the rules:

(MP): modus ponens, from $A \to B$ and A deduce B.
(UG): universal generalization, from $\vdash A(x)$ deduce $\vdash \forall x A(x)$.
(FG): future temporal generalization, from $\vdash A$ deduce $\vdash \Box_T A$.
(PG): past temporal generalization, from $\vdash A$ deduce $\vdash \Box_Y A$.

We will use the following axioms:

(1): any tautology of classical propositional logic with formulas of our language substituted for atoms.

[2]Again, if there are several modal operators, this rule is duplicated for each of them.

(2): quantifier axioms of classical predicate logic:
- **(2a):** $\forall x(A \to B) \to (A \to \forall xB)$, where x is not free in A.
- **(2b):** $\forall xA(x) \to A[t/x]$, where x does not appear free in the scope of a quantifier binding a variable from t.

(3T): $\Box_T(A \to B) \to (\Box_T A \to \Box_T B)$.
(3Y): $\Box_Y(A \to B) \to (\Box_Y A \to \Box_Y B)$.
(4T): $A \to \Box_T \Diamond_Y A$.
(4Y): $A \to \Box_Y \Diamond_T A$.
(5): $\Diamond_Y A \to \Box_Y A$.
(6): to ensure properties of the W predicate:
- **(6a):** $\exists x W(x)$.
- **(6b):** $W(x) \to \Box_Y^n \neg W(x) \wedge \Box_T^n \neg W(x)$, for all $n \geq 1$.
- **(6c):** $(W(x) \wedge A) \to \Box_Y^m \Box_T^n (W(x) \to A)$.
- **(6d):** $W(x) \wedge W(y) \to \Box_Y^m \Box_T^n (A(x) \leftrightarrow A(y))$.

(7): to ensure properties of the D predicate:
- **(7a):** $\Diamond_T^n W(x) \wedge \Diamond_T^n W(y) \to \Box_Y^p \Box_T^q D(x,y)$.
- **(7b):** $\Diamond_T^m W(x) \wedge \Diamond_T^n W(y) \to \Box_Y^p \Box_T^q \neg D(x,y)$, for $m \neq n$.

Axiom 5 guarantees that each node has a unique predecessor (the past is linear). Axiom 6a ensures that each world has a name, axiom 6b that two world on the same history will not have the same name, axiom 6c that two worlds with the same name contain the same formulas (we will show that we can then identify two such nodes in the completeness proof) and axiom 6d that if two elements of the domain are names for a single world then they are equivalent. Note that we do not need the Barcan axiom from modal logic, because we are in a symmetric setting (we can deduce it from axioms 4T and 4Y).

Let us note that a model where these axioms are valid is necessarily discrete: suppose that α, β and γ are nodes such that $\alpha < \beta < \gamma$ and $\alpha < \gamma$. If $W(y)$ holds in β, $\Diamond_Y W(y)$ holds in γ, and from axiom 5 we deduce that $\Box_Y W(y)$ also holds there. Therefore $W(y)$ holds in α. But this is impossible because of axiom 6b.

3.2. **Completeness proof.** Here is the overall completeness proof strategy. Typographical conventions in the rest of the paper: a greek lowercase letter for a node (e.g. α), a greek uppercase letter for a set of formulas (e.g. Φ), a roman uppercase letter for a formula, very often corresponding to the node it belongs to (e.g. $A \in \alpha$). In the expression above, α means both the node and the set of formulas it contains. When we say that α is consistent, we mean that the set of formulas contained in α is consistent.

We use a method akin to a tableau construction to build a model for a given formula F. As we are working in a quantified setting, we need to merge this tableau method with a Henkin type completion, so that in the end, for a given sentence A and a world α, either A or $\neg A$ holds in α.

We need an underlying structure to represent the model we are building. Let $\mathbb{N}$ be the set of all positive integers. To do this construction, we consider the set $T = \{0\}^*(\mathbb{N} \setminus \{0\})^*$, i.e. the set of all sequences of the form: any number of 0 followed by a finite sequence of non-zero natural numbers, as for example $(0,0,0,135,2,18)$. We write $\alpha\beta$ for the concatenation of two sequences α and β, and αb for the sequence obtained by appending the

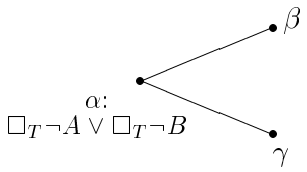

FIGURE 1

number b to the sequence α. We also write $\alpha a_1 \cdots a_p$, where the a_i are natural numbers and $p \geq 0$; by convention, if $p = 0$ this expression means α. We can define an order $<$ on this set by having $\alpha < \beta$ iff either $\beta = \alpha b$ where b is a non-zero natural number or $\alpha = 0\beta$. The ordered set $(T, <)$ thus obtained satisfies the "linear past, branching future" property. This set $(T, <)$ will be the skeleton of the model we are building. We will always be able to give a name to a node we need to build, be it in the past or in the future.

We also use an enumeration function *Enumerate* that gives us a sequence of tuples $(\alpha, A, code)$ where α is a node of T, A is a sentence and *code* is one of the following in the case of temporal logic:

- *add*: add either A or $\neg A$ to the node α.
- *build*$\Diamond_T$: (if A is of the form $\Diamond_T B$ and $A \in \alpha$) build a new β accessible from α containing B and $W(c)$ for a new constant c.
- *build*$\Diamond_Y$: (if A is of the form $\Diamond_Y B$ and $A \in \alpha$) if α has no yesterday, build a new yesterday node β containing B and $W(c)$ for a new constant c, else add A to the yesterday node of α.
- *exists*: (if A is of the form $\exists x A'(x)$ and $A \in \alpha$) add $A(c)$ for a new constant c.

All these actions are only to be accomplished if the node α has already been built. A tuple $(\alpha, A, code)$ is said to be *executable* if the node α has already been built and the preconditions (between brackets in the list above) are satisfied. In the sequence given by *Enumerate*, any tuple must appear infinitely often. At each step of our construction, we will execute a tuple if and only if it is possible. This *Enumerate* function enables us to build a sequences of finite trees with nodes containing sets of formulas. We must make sure that the model we build does not yield a contradiction when we start with the node denoted by the empty sequence ϵ containing a closed consistent formula. As we have said, we merge a Henkin completion with a tableau construction. In traditionnal Henkin completion, we know that it is always possible to add A or $\neg A$ to our set of sentences because if A cannot be added, then $\neg A$ can be deduced and we can add $\neg A$. Here however we add formulas to a node of a tree. Making sure that the set of formulas of a node is consistent is not enough, because the situation shown in Figure 1 could happen, where it might be locally consistent to add A in α and B in β, but this is "forbidden" by the formula $\Box_T \neg A \vee \Box_T \neg B$ in γ. Thus we must have a property of "global consistency" satisfied by the tree at each

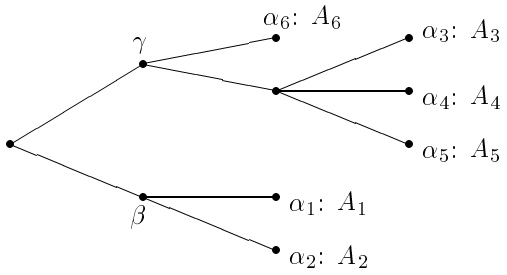

FIGURE 2

step. We begin by introducing path formulas. As we only build nodes when the $build\Diamond_T$ or $build\Diamond_Y$ codes arise, giving a successor or a predecessor to an already existing node, it is clear that two given nodes always have a closest common ancestor, more precisely for two nodes α and β, there exists γ, ancestor of α and β, such that for any δ ancestor of α and β, δ is an ancestor of γ.

Definition 3.1. Consider two nodes α and β, and let γ be their closest common ancestor. We therefore have $\alpha = \gamma a_1 \dots a_p$ and similarly $\beta = \gamma b_1 \dots b_q$. We will call *path formula* from α to β for a formula B the formula $\Diamond_Y^p \Diamond_T^q B$. We write $Path_{\alpha,\beta} B$ for such a formula.

Definition 3.2. A tree is said to be *globally consistent* if every node is consistent after doing any finite sequence of the following actions:

- adding the formula A to α if $\alpha \vdash A$.
- given two nodes α and β and $A \in \alpha$, adding $Path_{\beta,\alpha} A$ to α.

We will call such actions *conservative actions*. When a (finite) sequence of conservative actions enables us to add a formula to a node, we also say that we can add it conservatively.

Thanks to axioms 4T and 4Y, it is equivalent to allow the addition of path formulas or the propagation of $\Box$-formulas. It is obvious that if we have a globally consistent tree and if we perform any finite sequence of conservative actions, then the resulting tree is still globally consistent.

In order to use a Henkin type argument we need to be able to simulate in one node any sequence of conservative actions that could be applied to the tree. This is the purpose of *configuration formulas*, which represent in a way the structure of a finite part of the tree. We will only give here an example of a configuration formula and the properties we need for the completeness proof. The precise definition and proofs — which are rather tedious despite the intuitive ideas they reflect — can be found in appendix A. For a given number of nodes and formulas associated with these nodes, we wish to express the presence of these formulas in the tree without losing the information of their relative positions.

Consider for example Figure 2. It is not enough to add formula (3.1) to γ, because we lose the structure information. For example γ would not then be able to deduce $\Diamond_Y\Diamond_T(\Diamond_T A_1 \wedge \Diamond_T A_2)$. The configuration formula we are going to use is formula (3.2).

$$\Diamond_Y\Diamond_T^2 A_1 \wedge \Diamond_Y\Diamond_T^2 A_2 \wedge \Diamond_T^2 A_3 \wedge \Diamond_T^2 A_4 \wedge \Diamond_T^2 A_5 \wedge \Diamond_T A_6 \tag{3.1}$$

$$\Diamond_Y\Diamond_T(\Diamond_T A_1 \wedge \Diamond_T A_2) \wedge \Diamond_T(\Diamond_T A_3 \wedge \Diamond_T A_4 \wedge \Diamond_T A_5) \wedge \Diamond_T A_6 \tag{3.2}$$

We write $Conf_\gamma(\alpha_1 : A_1; \ldots; \alpha_6 : A_6)$ for this configuration formula.

The main result we obtain from using configurations is the following lemma, proved as lemma A.11 in the appendix.

Lemma 3.3. *Let $\mathcal{T}$ be a tree, α and β two nodes and A a sentence such that $A \in \alpha$. Suppose that a finite sequence of conservative actions enables us to add B in β. Then there is a configuration formula we can conservatively add to α in the tree $\mathcal{T}'$, obtained from $\mathcal{T}$ by removing the formula A from the node α, so that we can then deduce in the node α of $\mathcal{T}'$ the formula $A \to Path_{\alpha,\beta}B$.*

From this we easily deduce the result we need to build our tree sequence.

Lemma 3.4. *Suppose that after adding a formula A to a node α we get a contradiction somewhere in the tree. We could then have conservatively added $\neg A$ to α in the original tree.*

Proof. We have a node β such that we can add $\bot$ to β. But then by Lemma 3.3 we can conservatively add $A \to \Diamond_Y^p\Diamond_T^q\bot$ in the original tree in α. But by generalization $\vdash \Box_Y^p\Box_T^q\top$, and hence we can deduce $\neg A$ in α. □

We will now show that each tree in the sequence we are building is globally consistent.

Lemma 3.5. *If we have an enumeration function Enumerate and a globally consistent tree $\mathcal{T}$, after executing any executable tuple the resulting tree is globally consistent.*

Proof. For all the different cases of an executable tuple, we suppose that the tree is globally consistent at that stage and show that we can execute the tuple so that it remains globally consistent.

add: suppose that if we add A to α we get a contradiction after a finite sequence of actions, then by Lemma 3.4 we could have deduced $\neg A$ in α. We can therefore add $\neg A$ to α.

build$\Diamond_T$: if we build a new node β containing B and get a contradiction, it means that there is a configuration formula relative to α in the tree with the new node β containing B that enables us to get this contradiction. But because the formula $\Diamond_T B$ is in α in the original tree, we can add this configuration formula to α in the original tree, and thus we would get a contradiction. We can therefore build β containing B. If we cannot add $W(c)$ there for a new constant c, it means by lemma 3.4 that we can deduce $\neg W(c)$ in β. But then as the constant c does not appear anywhere else, we can get a proof of $\forall x \neg W(x)$ by Lemma A.2, but that is impossible because of axiom 6a.

build$\Diamond_Y$: if α already has a predecessor, then using axiom 5 we deduce $\Box_Y B$, and we can add B to that node. If it does not have a predecessor, then we reason as above.

exists: if we cannot add $A'(c)$ for a new constant c in α, we can then deduce $\forall x \neg A'(x)$ by Lemma A.2, which is incompatible with the fact that $\exists x A'(x) \in \alpha$. □

We can now prove the completeness of these axioms.

Theorem 3.6 (Completeness)**.** *Our axiomatization is complete if we consider the class of discrete linear past branching future models with locality and same-depth predicates.*

Proof. We begin the construction of our model with the formula A in the node ϵ. This tree is clearly globally consistent.

We then have a sequence of trees $\mathcal{T}_0, \ldots, \mathcal{T}_i, \ldots$ which are globally consistent. Call α^i the set of formulas contained in the node α of tree $\mathcal{T}_i$. Consider the structure $\mathcal{T}$with a node α containing $\bigcup_{i \geq 0} \alpha^i$. Note that for a sentence A, either $A \in \alpha$ or $\neg A \in \alpha$, because there is a tree $\mathcal{T}_i$ where we execute the tuple (α, A, add). Note also that in each of these nodes, any path sentence that could be added has been added and any sentence that could be deduced also. Indeed, if a sequence of actions could enable us to add a formula A to a node α, then there is a tree $\mathcal{T}_i$ where that sequence is possible, and the fact that all the successive trees are globally consistent means that we cannot have added $\neg A$, therefore $A \in \alpha$.

As domain for our model we will take the set of closed terms of the language considered, with all the new constants introduced, quotiented by equality if necessary.

Let us now show that if we consider our tree as a model, a sentence A is true at time α iff $A \in \alpha$. We use this property to define the truth of atomic sentences, and show that it is true for all other sentences by induction (we will only treat the cases of $\neg A'$, $A' \wedge A''$, $\exists x A'(x)$, $\Diamond_T A'$ and $\Diamond_Y A'$):

- if A is $\neg A'$, then A is true at α iff A' is not true at α, iff $A' \notin \alpha$ iff $\neg A' \in \alpha$.
- if A is $A' \wedge A''$, then A is true at α iff A' is true at α and A'' is true at α, iff $A' \in \alpha$ and $A'' \in \alpha$, iff $A' \wedge A'' \in \alpha$.
- if A is $\exists x A'(x)$, then:
  - if $A \in \alpha$, we have added $A'(c)$ for a given constant c during the construction, and by the induction hypothesis, $A'(c)$ is true at α, so that $\exists x A(x)$ is true at α.
  - if $A \notin \alpha$, $\neg \exists x A'(x) \in \alpha$, and therefore for any closed term t by deduction $\neg A'(t) \in \alpha$ and by induction hypothesis $A'(t)$ is not true at α, so $\exists x A'(x)$ is not true at α.
- if A is $\Diamond_T A'$:
  - if $A \in \alpha$, then at some point we have built a tomorrow node β containing A', so that by induction hypothesis, A' is true at β and A is true at α.
  - if A is true at α, then we have a tomorrow node β such that A' is true at β, hence by induction hypothesis $A' \in \beta$, and the path formula $\Diamond_T A'$ is of course in α, so that $A \in \alpha$.

- if A is $\Diamond_Y A'$:
  - if $A \in \alpha$, then there is a yesterday node β for α (either because we built it containing A' or because it was already there, in which case it also contains the path formula $\Diamond_T \Diamond_Y A'$, and by deduction A'), and by induction hypothesis A' is true at β so A is true at α.
  - if A is true at α, then we have a yesterday node β such that A' is true at β, hence by induction hypothesis $A' \in \beta$, and the path formula $\Diamond_Y A'$ is of course in α, so that $A \in \alpha$.

Now suppose that there are two distinct nodes α and β and a constant a such that $W(a) \in \alpha$ and $W(a) \in \beta$. Then using the axiom 6c we can prove that the nodes α and β contain the same formulas. Being distinct and not on the same history due to axiom 6b, α and β each have a yesterday node each, call these α' and β'. Then if $A' \in \alpha'$, we have the path formula $\Diamond_Y A'$ in α and hence in β. But we also have $\Box_Y A'$ in β by axiom 1, and then the path formula $\Diamond_T \Box_Y A'$ in β', and by deduction $A' \in \beta'$. We can thus prove that the nodes α' and β' contain the same formulas. We can reason in this manner until we reach the closest common ancestor γ of α and β. α and β were necessarily of same depth, else we would have two points on the same history with the same name, which is forbidden by axiom 6b. We therefore identify pairwise α and β and all their ancestors up to γ. In the tree we now have, two different nodes have different "names". Axiom 6d ensures that if two constants name one world, then they satisfy the same formulas.

Suppose finally that two worlds α and β, named by the constants a and b, have the same depth. Consider γ their closest common ancestor. There exists n such that the formulas $\Diamond_T^n W(a)$ and $\Diamond_T^n W(b)$ are in γ. The formula $D(a, b)$ is therefore in all worlds. If on the other hand α and β have different depths, there exists $m \neq n$ such that the formulas $\Diamond_T^m W(a)$ and $\Diamond_T^n W(a)$ are in γ. The formula $\neg D(a, b)$ is therefore in all worlds.

We have not ensured that there is a one to one correspondence between the domain and the set of worlds. If a world has several names (if $W(a)$ and $W(b)$ hold), we know thanks to axiom 6d that a and b satisfy the same formulas, so we can define an equivalence relation on the domain and factor those elements out. If we have a countably infinite number of worlds, we have a countably infinite number of elements, and this can be obtained by using a bijective application between the two sets to rename worlds. We will not worry too much in this section about this problem, because it will disappear in the transitive case. □

The construction above was started with node ϵ containing a single consistent formula. But we can do the same construction starting with a consistent theory Φ in node ϵ, because even if we do not have a finite structure at each step anymore, at each step we have only added a finite number of nodes containing a finite number of formulas to the original tree $\epsilon : \Phi$. We can therefore always find a new constant c when we need one. As before, contradictions in the tree involve a finite number of nodes and a finite number of formulas, so our reasoning is still valid. The main difference is that before, a configuration formula could represent the whole tree, and we could deduce everything we could have deduced in the tree from this one formula, whereas now we cannot represent the whole tree with one formula. But we

do not use this fact in our completeness proof. We thus obtain at no extra cost completeness for a consistent theory. This is also true of the other completeness results in this article.

4. Other logics

4.1. **Transitive temporal logic.** We again use two temporal operators $\Diamond_F$ and $\Diamond_P$, but they are transitive this time.

4.1.1. *The axioms.* We keep the rules (MP, UG, FG, PG), and basic axioms (1 to 4) from the preceding section (4T and 4Y are adapted and renamed 4F and 4P), and add the following:

(5'): $\Diamond_P A \wedge \Diamond_P B \to \Diamond_P(A \wedge B) \vee \Diamond_P(A \wedge \Diamond_P B) \vee \Diamond_P(\Diamond_P A \wedge B)$.
(TransF): to express the transitivity of $R_{\Box_F}$: $\Box_F A \to \Box_F \Box_F A$.
(TransP): to express the transitivity of $R_{\Box_P}$: $\Box_P A \to \Box_P \Box_P A$.
(6'): to ensure properties of the W predicate:
- **(6a'):** $\exists x W(x)$.
- **(6b'):** $W(x) \to \Box_F \neg W(x) \wedge \Box_P \neg W(x)$
- **(6c'):** $(W(x) \wedge A) \to \Box_P \Box_F (W(x) \to A) \wedge \Box_F (W(x) \to A) \wedge \Box_P (W(x) \to A)$.
- **(6d'):** $W(x) \wedge W(y) \to \Box_P \Box_F (A(x) \leftrightarrow A(y)) \wedge \Box_F (A(x) \leftrightarrow A(y)) \wedge \Box_P (A(x) \leftrightarrow A(y)) \wedge (A(x) \leftrightarrow A(y))$.
- **(6e'):** $\forall x (W(x) \vee \Diamond_P \Diamond_F W(x) \vee \Diamond_P W(x) \vee \Diamond_F W(x))$

Note that we now have access to all the worlds without using an infinite number of axioms, thanks to $\Box_P$ and $\Box_F$. Axiom 6e' for instance ensures that all elements of the domain are the name of a world somewhere.

We have not yet included axioms for the same-depth predicate, because such a predicate relies on tomorrow and yesterday nodes. We will nevertheless be able to add it due to the expressive power of the combined transitive operators and locality predicate.

4.1.2. *The Since and Until operators.* The operators Since (S) and Until (U) have the following meaning: $S(A, B)$ is true iff there is a time point in the past where A was true and B was true between that point and now; the converse condition holds for $U(A, B)$.
Formally, their satisfaction conditions are:

- $\mathcal{M}, \alpha, v \models U(A, B)$ iff $\exists\ \beta$ such that $\alpha < \beta$, $\mathcal{M}, \beta, v \models A$ and for all γ, if $\alpha < \gamma < \beta$, then $\mathcal{M}, \gamma, v \models B$.
- $\mathcal{M}, \alpha, v \models S(A, B)$ iff $\exists\ \beta$ such that $\beta < \alpha$, $\mathcal{M}, \beta, v \models A$ and for all γ, if $\beta < \gamma < \alpha$, then $\mathcal{M}, \gamma, v \models B$.

Now define $U(A, B)$ as the formula of our language:

$$\exists x (\Diamond_F (W(x) \wedge A) \wedge \Box_F (\Diamond_F W(x) \to B)).$$

This expresses the fact that there is a point β in the future where A holds and for any point in the future of now, if it is in the past of β then B holds. We get the converse definition for the $S(A, B)$:

$$\exists x (\Diamond_P (W(x) \wedge A) \wedge \Box_P (\Diamond_P W(x) \to B)).$$

4.1.3. *$\Diamond_T$ and $\Diamond_Y$ operators.* Now that we have the operators Since and Until we can use their expressive power to define the $\Diamond_T$ and $\Diamond_Y$ operators we used in the first logic we defined. Indeed, we have the following definitions:

$$\Diamond_T A \equiv U(A, \bot)$$

$$\Diamond_Y A \equiv U(A, \bot)$$

$U(A, \bot)$ states that there is a point in the future where A holds and $\bot$ holds at any intermediary point. But $\bot$ cannot hold anywhere, hence there are no intermediary points. A holds in an immediate successor of the current node, i.e. its tomorrow node. Let us write the precise expression of $\Diamond_T$ and $\Diamond_Y$.

$$\Diamond_T A \equiv \exists x(\Diamond_F(W(x) \wedge A) \wedge \neg\Diamond_F\Diamond_F W(x))$$
$$\Diamond_Y A \equiv \exists x(\Diamond_P(W(x) \wedge A) \wedge \neg\Diamond_P\Diamond_P W(x))$$

It is interesting to note that these formulas express that there is a point in the future (respectively the past) that cannot be reached in two steps, but only in one step, where A holds. In other words it means that a node is a tomorrow (respectively yesterday) node for another if it is in its future (respectively past) and there is no node in-between. We will now show that the operators we obtain behave as we expect them to.

Lemma 4.1. *Using the proposed axioms we obtain:*

$$\vdash \Diamond_Y A \to \Box_Y A$$
$$\vdash \Box_T(A \to B) \to (\Box_T A \to \Box_T B)$$
$$\vdash \Box_Y(A \to B) \to (\Box_Y A \to \Box_Y B)$$
$$\vdash A \to \Box_T\Diamond_Y A$$
$$\vdash A \to \Box_Y\Diamond_T A$$

Proof. Let us prove that $\Diamond_Y A \wedge \Diamond_Y B \to \Diamond_Y(A \wedge B)$. We will then deduce our first axiom by taking $B \equiv \neg A$. If we have $\Diamond_Y A \wedge \Diamond_Y B$, we have (4.1).

$$\exists x \exists y(\Diamond_P(W(x) \wedge A) \wedge \neg\Diamond_P\Diamond_P W(x) \wedge \Diamond_P(W(y) \wedge B) \wedge \neg\Diamond_P\Diamond_P W(y)) \tag{4.1}$$

Using axiom 5' on $\Diamond_P(W(x) \wedge A) \wedge \Diamond_P(W(y) \wedge B)$ we would get a disjunction of three different formulas, but $\neg\Diamond_P\Diamond_P W(x) \wedge \neg\Diamond_P\Diamond_P W(y)$ prohibits two of these, and we finally get (4.2).

$$\exists x \exists y(\Diamond_P(A \wedge B \wedge W(x) \wedge W(y)) \wedge \neg\Diamond_P\Diamond_P W(x) \wedge \neg\Diamond_P\Diamond_P W(y)) \tag{4.2}$$

From this we can deduce $\Diamond_Y(A \wedge B)$.

Let us now show the second axiom. Suppose we have $\Box_T(A \to B)$ and $\Box_T A$. Then we have (4.3), and from it we can deduce (4.4), which is exactly the formula we wanted.

$$\forall x(\Diamond_F\Diamond_F W(x) \vee (\Box_F(W(x) \to A) \wedge \Box_F(W(x) \to (A \to B)))) \tag{4.3}$$

$$\forall x(\Diamond_F\Diamond_F W(x) \vee (\Box_F(W(x) \to B))) \tag{4.4}$$

The third axiom is similar.

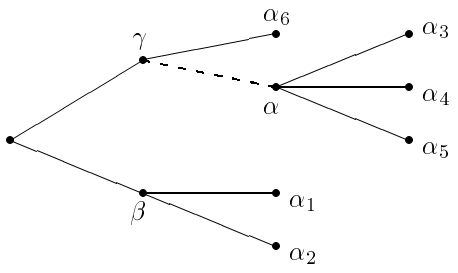

FIGURE 3

Let us show the fourth axiom. We will show the converse formula, i.e. $\Diamond_T \Box_Y A \to A$. $\Diamond_T \Box_Y A$ is formula (4.5).

$$(4.5)\quad \exists x(\Diamond_F(W(x) \wedge \forall y(\Diamond_P \Diamond_P W(y) \vee \Box_P(W(y) \to A))) \\ \wedge \neg\Diamond_F \Diamond_F W(x))$$

Using axiom 6a', we know that $\exists z W(z)$. We can use the $\forall x$ in the original formula and substitute x by z. We finally get formula (4.6).

$$(4.6)\quad \exists x \exists z(\Diamond_F(W(x) \wedge (\Diamond_P \Diamond_P W(z) \vee \Box_P(W(z) \to A))) \\ \wedge \neg\Diamond_F \Diamond_F W(x) \wedge W(z))$$

From $W(z) \wedge \neg\Diamond_F\Diamond_F W(x)$ we deduce via axiom 6c' $\Box_F\Box_P\Box_P(W(z) \to \Box_F\Box_F\neg W(x))$. We can use this to "bring in" $W(z) \to \Box_F\Box_F\neg W(x)$ with $W(z)$ in $(\Diamond_P\Diamond_P W(z) \vee \Box_P(W(z) \to A)$.
From this we deduce $(\Diamond_P\Diamond_P\Box_F\Box_F\neg W(x) \vee \Box_P(W(z) \to A)$, and then using axiom 4P, we get $\Box_P(W(z) \to A)$. If we replace all this in context, using axiom 4F we deduce $W(z) \wedge W(z) \to A$, and then A.

The fifth axiom is similar. □

4.1.4. *The same-depth predicate.* Having added tomorrow and yesterday operators, we can remember the same-depth predicate which we had defined in our first system where we only had $\Diamond_T$ and $\Diamond_Y$. Let us add to the axioms of section 4 the following:

(7a'): $\Diamond_T^n W(x) \wedge \Diamond_T^n W(y) \to D(x,y)$, for all n.
(7b'): $\Diamond_T^m W(x) \wedge \Diamond_T^n W(y) \to \neg D(x,y)$, for all $m \neq n$.
(7c'): $D(x,y) \to \Box_P D(x,y) \wedge \Box_F D(x,y) \wedge \Box_P\Box_F D(x,y)$

Note that with the axioms from 4.1.1 we do not force the model to have a (transitive) accessibility relation which is the transitive closure of a discrete accessibility relation. Indeed we could have a node α such that for all $\beta > \alpha$ there exists a node γ which is in-between ($\alpha < \gamma < \beta$). In that case, the node α has no tomorrow node. The same-depth axioms we have introduced link nodes that have the same discrete depth from a given node. Consider for example Figure 3. γ and β have same depth, α_3, α_4 and α_5 have same depth, α_6, α_1 and α_2 have same depth, but α does not have the same depth as α_6 or α_1, because of the gap between γ and β. If we wished to consider

discretely based models, we could add the following axioms (and similar ones for the past):

(Discrete): $W(x) \wedge \Diamond_F W(y) \rightarrow \exists z \Diamond_T (W(z) \wedge \Diamond_F W(y)) \vee \Diamond_T W(y)$
(Induction): $W(x) \rightarrow \forall y (\Box_F A(n(x,y)) \wedge$
$\forall z (\Diamond_F W(y) \wedge A(z) \rightarrow \Box_F A(n(z,y))) \rightarrow \Box_F A(y))$

In these axioms, $n(x,y)$ is the (unique) next time point after x on the history that leads to y. We need not introduce a new function symbol for it, because the relation $z = n(x,y)$ can be defined using an expression akin to the Discrete axiom. These axioms do not force the model to be discrete, but they ensure that such a discrete model exists.

If we do not wish to restrict ourselves to discrete models, the above axioms might not be precise enough. Indeed, in a model of these axioms with the structure of Figure 3, the D predicate could very well state that α and β have same depth, although one is separated from the root by a gap while the other is not. We can fortunately add axioms to ensure a more sensible behaviour for D. We say that there is there a gap between two nodes α and β ($\alpha < \beta$) if there is an infinite number of nodes between them. We therefore know that we have two sequences (α_i) and (β_i) of nodes such that:

$$\alpha \leq \alpha_0 \leq \cdots \leq \alpha_i \leq \cdots \leq \beta_i \leq \cdots \leq \beta_0 \leq \beta.$$

If one of these sequences converges towards a node of the model (i.e. for instance $\exists \gamma_0 \ \forall \gamma < \gamma_0, \ \exists i \ \gamma < \alpha_i \leq \gamma_0$), then there is a point between α and β which has no tomorrow node or no yesterday node on the history between α and β. We call such a gap a simple gap. Define $sgap(y)$ as the following formula:

$$\begin{aligned}(4.7)\quad & \neg\Diamond_T\Diamond_F W(y) \vee \neg\Diamond_F\Diamond_T W(y) \\ & \vee \Diamond_F(\Diamond_F W(y) \wedge \neg\Diamond_T\Diamond_F W(y) \wedge \neg\Diamond_T W(y)) \\ & \vee \Diamond_F(\Diamond_F W(y) \wedge \neg\Diamond_Y\Diamond_F W(y))\end{aligned}$$

This formula is true in a world of name x iff x has no tomorrow node leading to y, or y has no yesterday node coming from x, or there is a node between x and y which has no tomorrow node leading to y, or there is a node between x and y which has no yesterday node coming from x.

We add the following axiom:

(7d'): $\Diamond_F W(y) \wedge \Diamond_F W(z) \wedge \Box_F(\neg\Diamond_F W(y) \vee \neg\Diamond_F W(z))$
$\wedge (sgap(y) \vee sgap(z)) \rightarrow \neg D(y,z)$.
(7e'): $\Diamond_F(W(x) \wedge \Diamond_Y W(x_0)) \wedge \Diamond_F(W(y) \wedge \Diamond_Y W(y_0)) \wedge D(x,y)$
$\rightarrow D(x_0, y_0)$.
(7f'): $W(x) \wedge \Diamond W(y) \rightarrow \neg D(x,y)$.

The first one says that if x is the closest common ancestor of y and z and there is a simple gap between x and y or x and z, then x and y do not have same depth. The second one says if two nodes have same depth and each has a yesterday node, then those two yesterday nodes have same depth.

4.1.5. *Completeness proof.* To use the completeness proof of the previous case, we need to adapt the parts where we used axiom 5, because it has been replaced by axiom 5'. The following lemma, proved in appendix B as

lemma B.1, is a generalized version of axiom 5', will allow us to build nodes in the past.

Lemma 4.2. *Let $\alpha_0, \ldots, \alpha_n$ be nodes, $\Phi_1, \ldots, \Phi_n$ finite sets of formulas such that $\alpha_n < \cdots < \alpha_1 < \alpha_0$ then:*

$$\begin{aligned}\vdash\ \Diamond_P B \wedge Conf_{\alpha_0}(\alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \\ \rightarrow \bigvee_{1 \leq i \leq n} Conf_{\alpha_0}(\ldots; \alpha_i : \Phi_i \cup \{B\}; \ldots) \\ \vee \bigvee_{\substack{1 \leq i \leq n \\ \alpha_i < \beta < \alpha_{i-1} \\ \text{where } \beta \text{ is a new node}}} Conf_{\alpha_0}(\beta : B; \alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \\ \vee\ Conf_{\alpha_0}(\alpha_1 : \Phi_1; \ldots; \alpha_{n-1} : \Phi_{n-1}; \alpha_n : \Phi_n \cup \{\Diamond_P B\}).\end{aligned}$$

The configuration and simulation lemmas still work. In Lemma 3.5, when we build a past node to satisfy a $\Diamond_P B$ formula, we use our new Lemma 4.2 instead of axiom (5). Finally, we need to make sure that $(T, <)$ is an acceptable transitive time flow. Its transitive property is due to axioms (TransF) and (TransP). The problem is that we have identified nodes, and we cannot use axiom (5) anymore to identify all their ancestors. Instead, we need to make sure that we can indeed identify two given nodes. with same names.

Lemma 4.3. *Suppose we have nodes $\alpha_0, \ldots, \alpha_m$ and $\beta_1, \ldots, \beta_n$ such that $\alpha_0 < \alpha_1 < \cdots < \alpha_m$ and $\alpha_0 < \beta_1 < \cdots < \beta_n$, and for an element a, $W(a)$ holds both in α_m and β_n.*
If $m \geq 2$ and $n \geq 2$, there exist i and j and an element b such that $1 < i < m$ and $1 < j < n$ and $W(b)$ holds in both α_i and β_j.

Proof. We know that α_m and β_n contain the same formulas. Moreover, there exists an element b such that $W(b)$ holds in β_{n-1}. We therefore have $\Diamond_P W(b)$ in β_n, and in α_m. From our construction, we now know that $W(b)$ holds in a node in the past of α_m. It cannot be in α_0 or in the past of α_0, because those nodes are on the same history as β_{n-1} and axiom (6b') prohibits two nodes of the same history from having the same name. We therefore obtain the desired node α_i. □

Now using this lemma, we can identify nodes until we have either $m = 1$ or $n = 1$, and the resulting structure will yield an acceptable time flow.

The last point is to use our additionnal same-depth axioms to show properties of the D predicate. Thanks to the original axioms, inside a maximal discrete subtree the D predicate links nodes as expected. Thanks to axiom 7d', Two such subtrees separated by a simple gap have no elements of same depth. The last case is that of a gap which is not a simple gap. Let α be a node on a maximal discrete subtree and β a node from an other discrete subtree. Let γ be the closest common ancestor of α and β. Suppose that there at least one non simple gap between γ and α (and no simple gaps, because if there was one, α and β would be cleanly separated and it would be over). Suppose that $D(a, b)$ holds, where a and b are names for α and β. There cannot be a simple gap between γ and β, because then with axiom 7d' we would have $\neg D(a, b)$. Suppose there is no gap. Then there is a finite

number of nodes between γ and β. On the other hand, we can form a chain of yesterday nodes starting from α as long as we wish, because of the non simple gap. Thus at some point we would obtain using axiom 7e' that γ and some node between γ and α have same depth, which is impossible. In the model we obtain, D behaves as expected inside a discrete subtree, distinguishes discrete subtrees which are separated by simple gaps. It might however link discrete subtrees who are both separated from their closest common ancestor by a non simple gap.

We can now state our completeness result (we here get a one to one correspondence between worlds and elements).

Theorem 4.4. *Our axiomatization is complete for the class of all transitive linear past branching future models with locality and same depth predicate.*

4.2. **Modal logic.** In a modal setting, there is no notion of linear past, therefore one of the main differences between the two cases in temporal logic disappears. Many results will therefore be valid whether we are dealing with the transitive case or the non-transitive case.

We use exactly the logic defined in section 2, which is the usual definition of syntax and semantics. Note that we now use the Barcan formula.

We begin by giving axioms for the locality predicate, and then introduce axioms for the same-depth predicate.

4.2.1. *The axioms and rules.* In the transitive case, we can define an operator $\Diamond_1 A$ (1 stands for one step) similarly to $\Diamond_T$ in the previous section as $\exists x \Diamond (W(x) \wedge A) \wedge \neg \Diamond \Diamond W(x)$.

We keep the basic rules and axioms, and add the following:

- in the non-transitive case:
  (2a"): $\exists x W(x)$.
  (2c"): $W(x) \wedge A \rightarrow \Box^n (W(x) \rightarrow A)$, for all n.
  (2d"): $\Diamond^n (W(x) \wedge W(y)) \rightarrow \Box^m (A(x) \leftrightarrow A(y))$, for all m, n
  (2e"): $\Diamond^n W(x) \rightarrow \Box^m \neg W(x)$, for $m \neq n$.
  (2f"): $\Diamond^n (A \wedge \Diamond^m W(x)) \rightarrow \Box^n (\Diamond^m W(x) \rightarrow A)$.
- in the transitive case:
  (Trans): $\Box A \rightarrow \Box\Box A$.
  (2a"): $\exists x W(x)$.
  (2c"): $W(x) \wedge A \rightarrow \Box (W(x) \rightarrow A)$.
  (2d"): $(W(x) \wedge W(y)) \vee \Diamond (W(x) \wedge W(y)) \rightarrow (A(x) \leftrightarrow A(y)) \wedge \Box (A(x) \leftrightarrow A(y))$.
  (2e"): $\Diamond_1^n W(x) \rightarrow \Box_1^m \neg W(x)$, for $m \neq n$.
  (2f"): $\Diamond_1^n (A \wedge \Diamond_1^m W(x)) \rightarrow \Box_1^n (\Diamond_1^m W(x) \rightarrow A)$.
  (2g"): $W(x) \rightarrow \Box \neg W(x)$.

Note that even in the transitive case we lose the ability to access the whole tableau from an arbitrary node, we therefore have no equivalent of axiom (7'), which ensured that any element of the domain was the name of a world.[3] We solve this problem by building a model of $F \wedge \forall x \Diamond W(x)$ (or $\Phi \cup \{\forall x \Diamond W(x)\}$ in the case of a theory) in the transitive case, instead

[3]In essence, the usefulness of having access to all the worlds is linked to the notion of a universal modality; for instance, such a modality greatly simplifies the completeness proof in [3].

of building a model of just F, because the model we build starts from this node, thus for all x, there will be a world named by x.

For the same-depth predicate, we add the following in both cases:

(7a"): $\Diamond_1^n W(x) \wedge \Diamond_1^n W(y) \rightarrow D(x,y)$.
(7b"): $\Diamond_1^m W(x) \wedge \Diamond_1^n W(y) \rightarrow \neg D(x,y)$, for $m \neq n$.
(7c"): $D(x,y) \vee \Diamond D(x,y) \rightarrow D(x,y) \wedge \Box D(x,y)$.

We also need to add axioms in the transitive case, which are similar to the temporal case:

(7d'): $\Diamond W(y) \wedge \Diamond W(z) \wedge \Box(\neg\Diamond W(y) \vee \neg\Diamond W(z)) \wedge (sgap(y) \vee sgap(z)) \rightarrow \neg D(y,z)$.
(7e'): $\Diamond(W(x_0) \wedge \Diamond_1 W(x)) \wedge \Diamond(W(y_0) \wedge \Diamond_1 W(y)) \wedge D(x,y) \rightarrow D(x_0,y_0)$.

In these axioms, sgap is the adaptation of the formula defined in the temporal transitive case.

4.2.2. *Completeness proof.* We will start by adapting the notions we have developed for temporal logic in the modal case.

This is an obvious statement, but a node can only access nodes to which it is linked by the accessibility relation. In a certain sense, it is as if we were in a temporal logic, but with only future operators. A node can only access its successors. This means that a configuration from a point γ to nodes $\alpha_1, \ldots, \alpha_n$ is only possible if all the nodes are successors, where β is a successor of α iff there are nodes $\alpha_0, \ldots, \alpha_n$, $(n \geq 1)$ such that $\alpha = \alpha_0 < \cdots < \alpha_n = \beta$. Similarly, a path formula $Path_{\alpha,\beta}B$ is only possible if β is a successor of α. We modify our definitions of path and configuration formulas accordingly in a straightforward way.

Because a configuration can only access the successors of its base point, we will reason in the root ϵ of our tree during the construction. Indeed, the root is the only node that can "see" all the information which is in the tree. We could not have done this in the case of temporal logic because we could build nodes in the past. The set we now consider as a skeleton is just the set of sequences of natural numbers, on which we define the usual order to get a tree where each node may have an infinite number of sons, but of course only one father.

We also need to modify the notion of global consistency. It is no longer equivalent to allow the propagation of $\Box$-formulas and the adding of path formulas. Recall Figure 1: if we just allow the propagation of $\Box$-formulas, we can still add A in α and B in β.

Definition 4.5. A tree is said to be *globally consistent* if every node is consistent after doing any finite sequence of the following actions:

- adding the formula A to α if $\alpha \vdash A$.
- given two nodes α and β such that β is a successor of α and $A \in \alpha$, adding $Path_{\beta,\alpha}A$ to α.
- given two nodes α and β such that β is a successor of α and $\Box^n \in \alpha$, adding A to β.

Our configurations can be seen as a special case of the configurations defined in the temporal section. But we must make sure for each lemma that we are not using axioms that are strictly temporal (namely the fact

that we have operators to access the past and future and that they are the converse of each other).

We used temporal axioms to prove that we could add path formulas. We will therefore need to prove this result again. Moreover, in the temporal case, we did not propagate $\Box$-formulas because it was equivalent to add path formulas, but as we have seen it is now necessary. The necessary new results are proved in appendix C.

Finally, we need to prove the following lemma (see lemma C.6 in the appendix) because we will only be reasoning in the root ϵ of the tree. In the temporal case, when we wanted to show that we could add $A(c)$ for a new constant c, we would just use the classical result (Lemma A.2). But we now need to adapt this result so that we can use it from ϵ.

Lemma 4.6. *Let $\mathcal{T}$ be a tree and α be a node in $\mathcal{T}$. Let D and $A(x)$ be formulas, where x is free in A. Suppose we can conservatively add to α $A[c/x]$ for a new constant c. Then there exists a variable w, a finite set of formulas Φ contained in α and there are nodes $\alpha_1, \ldots, \alpha_n$ in the tree, containing finite sets of formulas $\Phi_1, \ldots, \Phi_n$ such that:*

$$\vdash Conf_\epsilon(\alpha : \Phi \cup \{D\}; \alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \rightarrow Path_{\epsilon,\alpha} D \wedge \forall w A[w/x].$$

If we now follow the same proof pattern, we just need to make a few comments. The code $build\Diamond_Y$ no longer exists, and $build\Diamond_T$ becomes $build\Diamond$. An adapted Lemma 3.5 still holds, because of the lemmas we have just shown. We are showing completeness for the class of all connected sets of worlds (or all transitive connected sets of worlds), therefore we can identify nodes without any anxiety.

In the non-transitive case, we do not ensure that any element of the domain is the name of a world, and the same remarks apply when we have an infinite countable number of worlds.

In the transitive case, we can once again solve this problem by building a model of the formula $F \wedge \forall x(W(x) \vee \Diamond W(x))$, instead of just building a model for F.

Once we have built a model, we use axioms 2e", 2f", 2g" to identify worlds and ensure that two different worlds do not share the same name. The argument for the same-depth predicate is similar here, and we obtain the same kind of properties.

Theorem 4.7. *Our axiomatizations are respectively complete for the class of all discrete models with locality predicate and the class of all transitive models with locality predicate and same-depth predicate.*

5. Simulating other logics: hybrid logics

The hybrid logics of Blackburn and Tzakova [2] are propositional logics with additional variables and constants that are used to name worlds, and quantifiers that bind these state variables. We will describe briefly these logics and show how they can be simulated using our transitive modal logic with locality predicate.

5.1. **A brief outline of syntax and semantics.** Two hybrid logics are defined: $\mathcal{L}(B)$ with $B \in \{\forall, \downarrow\}$. These are basically propositional languages, where the atomic symbols are sorted in three sets, assumed to be disjoint: PROP (the usual propositional symbols), SVAR (the state variables) and NOM (the state nominals). The well-formed formulas of $\mathcal{L}(B)$ are built inductively from the atomic symbols with the connectives $\neg, \wedge$, the operator $\Box$ and the quantifier B which binds the state variables of a formula.

The semantics are defined for a model $\mathcal{M}$ which is a triple (S, R, V), such that S is a non-empty set, R is a binary relation on S, and V : PROP $\cup$ NOM $\rightarrow Pow(S)$. Only standard valuations are considered, i.e. which map any nominal to a singleton subset of S. An assignment is a mapping g : SVAR $\rightarrow Pow(S)$. There is again a restriction to standard assignments, i.e. which map a state variable to a singleton subset of S. Let $[V, g](a) = g(a)$ if a is a state variable and $V(a)$ otherwise. The satisfaction condition for atoms is $\mathcal{M}, g, s \models a$ iff $s \in [V, g](a)$. Satisfaction is defined in the usual way for $\neg, \wedge, \Box$. The satisfaction of $\forall$ and $\downarrow$ is defined thus:

$$\mathcal{M}, g, s \models \forall x \phi \text{ iff for all } g' \overset{x}{\sim} g,\ \mathcal{M}, g', s \models \phi$$

$$\mathcal{M}, g, s \models\downarrow x \phi \text{ iff for all } g' \overset{x}{\sim} g, \text{ such that } g'(x) = s,\ \mathcal{M}, g', s \models \phi.$$

5.2. **Translation.**

5.2.1. *Structures.* Let $\mathcal{M} = (S, R, V)$ be a model, we define a model $\mathcal{M}^* = (T, R_\Diamond, \mathcal{D}, h)$ of our logic by taking $T = S$, our domain is $D = S$, $R_\Diamond = R$, h is defined for the propositional symbols p (i.e. the 0-ary predicates in our language) in a natural way, $\mathcal{D}$ contains the interpretations $V(i)$ of the nominals (which are constants in our language). We interpret the predicate W in the following way: $h(W, s) = \{s\}$.

Let g be an assignment, it is therefore defined on the set of state variables, which will be our set of variables, and maps them to worlds in the set S which is also our domain, so we define an assignment for our language: $g^* = g$.

5.2.2. *Formulas.* We now define for all formulas of $\mathcal{L}(\forall)$ a translation to a formula of our language. We translate $\Box$ to $\Box$, which is transitive, but if we want to simulate a discrete logic we just translate $\Box$ to the discrete operator we can define using $\Box$ in our logic and the locality predicate. The definition is by induction on its structure:

$$\begin{aligned}
[p]^* &= p, \text{ where } p \in \text{PROP} \\
[y]^* &= W(y), \text{ where } y \in \text{SVAR} \\
[i]^* &= W(i), \text{ where } i \in \text{NOM} \\
[\neg\phi]^* &= \neg[\phi]^* \\
[\phi \wedge \psi]^* &= [\phi]^* \wedge [\psi]^* \\
[\Box\phi]^* &= \Box[\phi]^* \\
[\forall x \phi]^* &= \forall x [\phi]^*
\end{aligned}$$

Note that we do not define a translation for $\downarrow$, although it would be possible, because it can be defined using $\forall$: $\downarrow x \phi \equiv \exists x (x \wedge \phi)$.

5.2.3. *Faithfulness of the translation.* We show by induction that for all formulas ϕ of $\mathcal{L}(\forall)$, $\mathcal{M}, g, s \models \phi$ iff $\mathcal{M}^*, g^*, s \models [\phi]^*$:

- if ϕ is atomic:
  - if it is a propositional symbol p, its translation is p, and the result is clear because we have the same set of worlds and the same extensions for propositional symbols.
  - if it is a nominal i, $\mathcal{M}, g, s \models i$ iff $s = V(i)$; its translation is $W(i)$, and $\mathcal{M}^*, g^*, s \models W(i)$ iff $s = V(i)$, because of the interpretation of the constant i and of the predicate W.
  - if it is a state variable y, $\mathcal{M}, g, s \models y$ iff $s = g(y)$; its translation is $W(y)$, and $\mathcal{M}^*, g^*, s \models W(y)$ iff $s = g(y)$, because of the definition of g^* and of the interpretation of the predicate W.
- the cases for $\neg, \wedge, \Box$ are straightforward.
- if it is of the form $\forall x\psi$, $\mathcal{M}, g, s \models \forall x\psi$ iff $\forall g'(g' \overset{x}{\sim} g \Rightarrow \mathcal{M}, g', s \models \psi$, iff $\mathcal{M}^*, g^*, s \models [\forall x\psi]^*$, because we have defined $g^* = g$.

6. Further developments

6.1. **Geographical predicates.** The same-depth predicate that we have defined is an example of the "geographical" predicates we can define once we have the locality predicate. If we observe the proofs we have done, the tools introduced are used to enable us to build a model with the locality predicate W. The axioms for D are just used to check that it has the desired properties. To define other predicates based on distance between nodes, one need just find an adequate set of axioms.

Let us for example consider the case of transitive temporal logic and define a countable number of predicates $(d_n(x))_{\mathbb{N}}$: their intended meaning is that x has depth n from the root. Consider the following axioms:

(D1): $\neg\Diamond_Y\top \wedge W(x) \rightarrow d_0(x)$.
(D2): $d_n(x) \wedge W(x) \wedge \Diamond_T W(y) \rightarrow d_{n+1}(y)$.
(D3): $d_n(x) \wedge W(x) \wedge \Diamond_Y W(y) \rightarrow d_{n-1}(y)$, for $n \geq 1$.
(D4): $d_0(x) \wedge W(x) \rightarrow \neg\Diamond_Y\top$.

This works well in a discretely based model. If there are gaps, only maximal discrete subtrees containing a root will have elements satisfaying a given d_n.

6.2. **Structured constants and substructural logics.** Because we are building the model "by hand", we have a certain amount of control over what we are adding. Consider the skeleton model we use for our construction; we could use it as a basis to add structured constants as names of worlds. The name of a node would give information on its place in the model. This would in turn enable us to simulate other logics such as substructural logics. In such a logic there is a different satisfaction condition for $\rightarrow$:

$$\mathcal{M}, \alpha, v \models A \rightarrow B \text{ iff for all } \beta,\ \mathcal{M}, \beta, v \models A \text{ implies } \mathcal{M}, \alpha * \beta, v \models B.$$

$*$ is the concatenation function. Our structure on names would allow us to express such a condition.

6.3. **Simulating other logics and defining new operators.** As we have seen the expressive power of a transitive operator with the locality predicate enables us to define the Since and Until operators. We could define some other operators. Consider for instance the LDS (Labelled Deductive Systems) presentation of temporal logic. It is used to define new bounded operators $G^x A$ and $H^x A$, where $G^x A$ means that for all nodes in the future between now and a point labelled x, A holds. If we also allow labelled operators we can define them: $G^x A \equiv U(W(x), A)$.

We should also be able to simulate the LDS presentation of temporal logic within our framework, because we can represent a configuration with our configuration formulas. Details about LDS can be found in [6].

Conclusion

We have introduced a locality predicate W in modal and temporal logic and given axioms to ensure its properties. As we have seen with the logic simulation and the other possible applications, it gives rise to powerful logics. We have noticed that it was then quite simple to define a same-depth predicate. The expressive power gained can also be illustrated by simulating other logics. There are many possible applications which have been but sketched. If we consider a given logic and a model of a given sentence, we can add new names to worlds, we then obtain a consistent theory, which has a model, where we can give different names to the new worlds. By iterating this process we obtain a model where all worlds have a name. Here instead we have built the models from a given axiomatization. We get a set of axioms and we can retain this construction to get further results, such as adding structured constants to name the worlds, because of the control we keep in the process.

Appendix A. Technical lemmas for the discrete temporal case

Let us first prove two classical results about proofs that we will be using quite often implicitly.

Lemma A.1. *Let Φ be a set of formulas, F and G formulas, if $\Phi \cup \{F\} \vdash G$, then $\Phi \vdash F \to G$.*

Proof. Let $(G_0, \dots, G_n)$ be a proof of G in $\Phi \cup \{F\}$, we will build a proof of $F \to G$ in Φ by inserting some formulas in the sequence $(F \to G_0, \dots, F \to G_n)$ so that it becomes a proof.
If G_i is a tautology, so is $F \to G_i$.
If G_i is an axiom or belongs to Φ, insert between $F \to G_{i-1}$ and $F \to G_i$ (or simply put before $F \to G_0$ if $i = 0$) the formulas G_i and $G_i \to (F \to G_i)$ (a tautology). If G_i is obtained by (MP), then we have $j, k < i$ with $G_k \equiv G_j \to G_i$. We insert between $F \to G_{i-1}$ and $F \to G_i$ the formulas $(F \to G_j) \to ((F \to (G_j \to G_i)) \to (F \to G_i))$ and $(F \to (G_j \to G_i)) \to (F \to G_i)$.
If G_i was obtained by generalization, then $\vdash G_i$ and we insert the tautology $G_i \to (F \to G_i)$. □

Lemma A.2. *Let Φ be a set of formulas, $F(x)$ a formula and c a constant not appearing in Φ nor F. If $\Phi \vdash F[c/x]$, then there exists a variable w such that $\Phi \vdash \forall w F[w/x]$.*

Proof. We therefore have $\bigwedge_{1 \le i \le n} F_i \vdash F[c/x]$, where $F_i \in \Phi$. Using the previous lemma we obtain $\vdash \bigwedge_{1 \le i \le n} F_i \to F[c/x]$. Let w be a variable not appearing in any of the F_i or in F. If we replace the symbol c by w, we get $\vdash \bigwedge_{1 \le i \le n} F_i \to F[w/x]$. By generalization and using axiom 2a, we deduce $\vdash \bigwedge_{1 \le i \le n} F_i \to \forall w F[w/x]$, and then $\Phi \vdash \forall w F[w/x]$. □

A.1. **The configuration lemmas.** We give here the precise definition of a configuration formula and prove basic properties.

Lemma A.3. *Let α, β and γ be three nodes such that the paths from α to β and α to γ have only α as a common point. Then the following formula can be deduced from the axioms:*

$$Path_{\alpha,\beta} B \wedge Path_{\alpha,\gamma} C \to Path_{\alpha,\beta}(B \wedge Path_{\beta,\gamma} C).$$

Proof. This is straightforward by case analysis on the possible positions of α, β and γ, and using axioms 4T and 4Y. □

We can now give a formal definition of a configuration formula.

Definition A.4. Let $\alpha_1, \dots, \alpha_n$ and γ be nodes, $\Phi_1, \dots, \Phi_n$ finite sets of formulas, $Conf_\gamma(\alpha_1 : \Phi_1; \dots; \alpha_n : \Phi_n)$ is defined by induction on n:

- if $n = 0$ by convention $Conf_\gamma(\emptyset)$ is $\top$.
- if $n = 1$ let $Conf_\gamma(\alpha_1 : \Phi_1) = Path_{\gamma,\alpha_1} \bigwedge_{F \in \Phi_1} F$.
- if $n \ge 2$, let β be the closest node to α_n on the path from γ to α_n that also belongs to a path leading from γ to an α_i; let $J \subseteq \{1, \dots, n-1\}$ be such that $j \in J$ iff α_j is in the future of β; then let $Conf_\gamma(\alpha_1 : \Phi_1; \dots; \alpha_n : \Phi_n)$ be the formula:

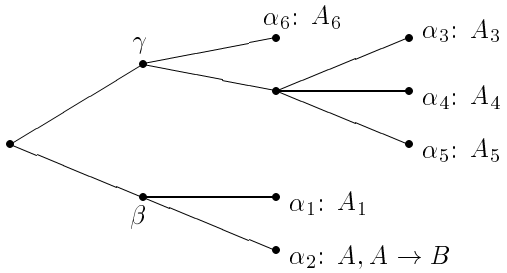

FIGURE 4

$$Conf_\gamma(\alpha_j : \Phi_j, j \notin J \cup \{n\};$$
$$\beta : \{(Path_{\beta,\alpha_n} \bigwedge_{F \in \Phi_n} F) \wedge Conf_\beta(\alpha_j : \Phi_j, j \in J)\}).$$

We call n the *degree* of the formula.

If $\Phi_i = \{F_i\}$, we will write $Conf_\gamma(\ldots;\alpha_i : F_i;\ldots)$ instead of the correct $Conf_\gamma(\ldots;\alpha_i : \{F_i\};\ldots)$. From now on, we will also often write $Path_{\alpha,\beta}\Phi$ when Φ is a finite set of formulas instead of $Path_{\alpha,\beta} \bigwedge_{F\in\Phi} F$. In the previous definition, the set J is non-empty and contains at most $n-1$ nodes, therefore the configuration formulas used are indeed defined. When we write $(\alpha_i : \ldots)$ in the inductive definition of the configuration formula, we mean the configuration with the sequence of nodes whose indices are not in $J \cup \{n\}$ in the order given by the original sequence $\alpha_1, \ldots, \alpha_n$, so that we get a unique formula in the definition. As we will see with Lemma A.6, the order in which the nodes are stated does not affect the meaning of the formula and from then on we shall consider any permutation as the configuration formula. We assume the same writing precautions in the proof of Lemma A.5 below.

As we want to use configuration formulas to simulate conservative actions, the least we can expect is that if we can deduce a formula from the formulas in a node, then we can deduce the resulting configuration formula from the original one. Consider for example Figure 4, we can deduce B in node α_2, therefore we want to be able to deduce $Conf_\gamma(\alpha_1 : A_1, \alpha_2 : B, \ldots, \alpha_6 : A_6)$ if we suppose the hypothesis $Conf_\gamma(\alpha_1 : A_1, \alpha_2 : \{A, A \to B\}, \ldots, \alpha_6 : A_6)$.

Lemma A.5. *Let $\alpha_1, \ldots, \alpha_n$ and γ be nodes, $\Phi_1, \ldots, \Phi_n$ finite sets of formulas, such that $\Phi_i \vdash G$, then we have:*

$$\vdash Conf_\gamma(\alpha_1 : \Phi_1;\ldots;\alpha_i : \Phi_i;\ldots;\alpha_n : \Phi_n)$$
$$\to Conf_\gamma(\alpha_1 : \Phi_1;\ldots;\alpha_i : G;\ldots;\alpha_n : \Phi_n).$$

Proof. If $\Phi_i \vdash G$, then using Lemma A.1, we get $\vdash \bigwedge_{F\in\Phi_i} F \to G$. We therefore just need to show that if we have $\vdash \bigwedge_{F\in\Phi_i} F \to G$, then

$$\vdash Conf_\gamma(\alpha_1 : \Phi_1; \ldots; \alpha_i : \Phi_i; \ldots; \alpha_n : \Phi_n) \\ \to Conf_\gamma(\alpha_1 : \Phi_1; \ldots; \alpha_i : G; \ldots; \alpha_n : \Phi_n).$$

The proof is achieved by induction on the degree n of the formula.

Case $n = 1$. The configuration formula is $Path_{\gamma,\alpha_1}\Phi_1$, and because for $\Diamond \in \{\Diamond_T, \Diamond_Y\}$ we can deduce $\vdash \Diamond F \to \Diamond G$ from $\vdash F \to G$, it is straightforward to show:

$$\vdash (Path_{\gamma,\alpha_1}\Phi_1) \to Path_{\gamma,\alpha_1} G.$$

Case $n \geq 2$. We use the inductive definition of the configuration formula $Conf_\gamma(\ldots; \alpha_i : \Phi_i; \ldots)$, which is formula (A.1).

$$Conf_\gamma(\alpha_j : \Phi_j, j \notin J; \beta : Conf_\beta(\alpha_j : \Phi_j, j \in J) \wedge Path_{\beta,\alpha_n}\Phi_n) \tag{A.1}$$

If $i \notin J$, then we can use the induction hypothesis on this configuration formula which is of a smaller degree. If $i \in J$, then we first get formula (A.2) by induction hypothesis. We show the desired result by another application of the induction hypothesis.

$$\vdash Conf_\beta(\alpha_j : \Phi_j, j \in J) \to Conf_\beta(\alpha_j : \Phi_j, j \in J\backslash\{i\}; \alpha_i : G) \tag{A.2}$$

If $i = n$, then we have as in the case $n = 1$:

$$\vdash (Path_{\beta,\alpha_n}\Phi_n) \to Path_{\beta,\alpha_n} G.$$

We then deduce formula (A.3) and use once more the induction hypothesis to end the proof.

$$\vdash (Path_{\beta,\alpha_n}\Phi_n) \wedge Conf_\beta(\alpha_j : \Phi_j, j \in J) \\ \to (Path_{\beta,\alpha_n} G) \wedge Conf_\beta(\alpha_j : \Phi_j, j \in J) \tag{A.3}$$

□

We now prove what we claimed just before: the order of the nodes is of no consequence. It means that we can "build" the configuration using the inductive case of the definition taking any of the nodes as the last node, the one which is a reference.

Lemma A.6. *Let σ be a permutation of $\{1, \ldots, n\}$, then we have:*

$$\vdash Conf_\gamma(\alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \\ \leftrightarrow Conf_\gamma(\alpha_{\sigma(1)} : \Phi_{\sigma(1)}; \ldots; \alpha_{\sigma(n)} : \Phi_{\sigma(n)}),$$

for all nodes $\alpha_1, \ldots, \alpha_n$ and γ and finite sets of formulas $\Phi_1, \ldots, \Phi_n$.

Proof. We will again prove this by induction on the degree n of the configuration formula.

Case $n = 1$. There is nothing to prove.

Case $n \geq 2$. We just need to show this for a transposition of the form $(1, i)$. If $i \neq n$, by using the definition of the configuration formula and the induction hypothesis we get the result. We will treat in more details the case of the transposition $(1, n)$. By definition, the original configuration formula

is formula (A.4), and similarly the configuration formula with nodes 1 and n transposed is formula (A.5).

(A.4) $Conf_\gamma(\alpha_j : \Phi_j, j \notin J_n \cup \{n\};$
$\beta_n : Conf_{\beta_n}(\alpha_j : \Phi_j, j \in J_n) \wedge Path_{\beta_n,\alpha_n}\Phi_1).$

(A.5) $Conf_\gamma(\alpha_j : \Phi_j, j \notin J_1 \cup \{1\};$
$\beta_1 : Conf_{\beta_1}(\alpha_j : \Phi_j, j \in J_1) \wedge Path_{\beta_1,\alpha_1}\Phi_1).$

First case: $n \notin J_1$ and $1 \notin J_n$. In that case, the sets J_1 and J_n are disjoint. Then in each of the above formulas ((A.4) and (A.5)) we can use the induction hypothesis to permute the orders and get equivalent formulas that share the order of their common nodes (the α_i such that $i \notin J_1 \cup J_n$), and in the first α_1 is the last node and in the second α_n is the last node. We then use the definition of a configuration formula again for each of them and get the same formula. The original formulas are therefore equivalent.

Second case: $n \in J_1$. Then β_n is in the future of β_1, and $J_n \subset J_1$. Again we use the induction hypothesis to get a reordered formula equivalent to (A.4) and where α_1 is the last node. If we then use the definition of this configuration formula, we get (A.6).

(A.6) $Conf_\gamma(\alpha_i : \Phi_i, i \notin J_1 \cup \{1\}; \beta_1 : (Path_{\beta_1,\alpha_1}\Phi_1)$
$\wedge Conf_{\beta_1}(\alpha_i : \Phi_i, i \in J_1 \backslash J_n; \beta_n : (Path_{\beta_n,\alpha_n}\Phi_n)$
$\wedge Conf_{\beta_n}(\alpha_i : \Phi_i, i \in J_n))).$

By induction hypothesis we reorder $Conf_{\beta_1}(\alpha_j : \Phi_j, j \in J_1)$, which appears in A.5, to get an equivalent formula where α_n is the last node and then use the definition of the resulting configuration formula to obtain (A.7).

(A.7) $Conf_{\beta_1}(\alpha_i : \Phi_i, i \in J_1 \backslash J_n;$
$\beta_n : (Path_{\beta_n,\alpha_n}\Phi_n) \wedge Conf_{\beta_n}(\alpha_i : \Phi_i, i \in J_n))$

Using Lemma A.5 we get the equivalence of our original formulas.

Third case: $1 \in J_n$. This is treated in he same way as the second case. □

Configuration formulas of different degrees can represent the same situation. In the example of Figure 2, we could also have written:

$$Conf_\gamma(\beta : \Diamond_T A_1 \wedge \Diamond_T A_2; \alpha_3 : A_3, \ldots, \alpha_6 : A_6).$$

We deal with the fact thus illustrated that we can replace a set of nodes in a configuration by one node containing a subconfiguration in the following lemma.

Lemma A.7. *Let $\alpha_1, \ldots, \alpha_n$, δ and γ be nodes, $\Phi_1, \ldots, \Phi_n$ finite sets of formulas. Suppose that δ is on a path from γ to an α_i, let J be the set of all indices j such that α_j is in the future of δ. Then we can prove from the axioms that the following formulas are equivalent.*

$$Conf_\gamma(\alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n)$$

$$Conf_\gamma(\alpha_i : \Phi_i, i \notin J; \delta : Conf_\delta(\alpha_j : \Phi_j, j \in J))$$

Proof. We proceed by induction on the degree n of the configuration formula.

Case $n = 1$. The result is easy to prove.

Case $n \geq 2$. In our definition of a configuration formula, if α_j is the last node, we get a node β_j. Let us choose k so that β_k is the closest to δ of all β_j that are in the future of δ. Using Lemma A.6 with α_k as our privileged point for the definition of the configuration formula yields (A.8). We wish to show the equivalence of formulas (A.8) and (A.9).

$$Conf_\gamma(\alpha_j : \Phi_j, j \notin J_k; \beta_k : Conf_{\beta_k}(\alpha_j : \Phi_j, j \in J_k) \wedge Path_{\beta_k,\alpha_k}\Phi_k) \tag{A.8}$$

$$Conf_\gamma(\alpha_i : \Phi_i, i \notin J; \delta : Conf_\delta(\alpha_j : \Phi_j, j \in J)) \tag{A.9}$$

By the choice of α_k, $J = J_k \cup \{\alpha_k\}$. The definition of the configuration formula $Conf_\delta(\alpha_j : \Phi_j, j \in J)$ in (A.9) is (A.10).

$$Conf_\delta(\beta_k : Conf_{\beta_k}(\alpha_j : \Phi_j, j \in J_k) \wedge Path_{\beta_k,\alpha_k}\Phi_k) \tag{A.10}$$

Lemma A.5 proves the equivalence of (A.9) and (A.11).

$$\begin{aligned} Conf_\gamma(&\alpha_i : \Phi_i, i \notin J; \\ &\delta : Conf_\delta(\beta_k : Conf_{\beta_k}(\alpha_j : \Phi_j, j \in J_k) \wedge Path_{\beta_k,\alpha_k}\Phi_k)). \end{aligned} \tag{A.11}$$

Finally we use the induction hypothesis on formula (A.9), and we get (A.8). □

The next lemma just shows that we can extract information from a configuration formula. This means that we can deduce from the original configuration formula a configuration formula by forgetting a node. By repeating this process, we can deduce a path formula from a configuration formula. Consider again Figure 2, then from the formula $Conf_\gamma(\alpha_1 : A_1, \ldots, \alpha_6 : A_6)$ we can successively forget the nodes $\alpha_2, \ldots, \alpha_6$ so that we deduce the path formula $Path_{\gamma,\alpha_1}A_1$.

Lemma A.8. *Let $\alpha_1, \ldots, \alpha_n$ and γ be nodes, $\Phi_1, \ldots, \Phi_n$ finite sets of formulas, then for all $i \in \{1, \ldots, n\}$ we have:*

$$\vdash Conf_\gamma(\alpha_j : \Phi_j, 1 \leq j \leq n) \rightarrow Conf_\gamma(\alpha_j : \Phi_j, 1 \leq j \leq n, j \neq i).$$

Proof. By induction on the degree n of the configuration formula.

Case $n = 1$. This is true because of our convention that an empty configuration formula is $\top$.

Case $n \geq 2$. We can then choose α_i to use the definition of the configuration formula, and we get:

$$Conf_\gamma(\alpha_j : \Phi_j, j \notin J \cup \{i\}; \beta : Conf_\beta(\alpha_j : \Phi_j, j \in J) \wedge Path_{\beta,\alpha_i}\Phi_i).$$

We get rid of the path formula relative to α_i with Lemma A.5, and we get the expected configuration formula using Lemma A.7. □

The second step in our simulation programme is to show that we can add a path formula to a node, another conservative action we wish to simulate. In the tree of Figure 2, we can add the formula $\Diamond_Y\Diamond_T A_2$ to the node α_1. We show in the following lemma that we can then deduce $Conf_\gamma(\alpha_1 : \{A_1, \Diamond_Y\Diamond_T A_2\}, \ldots, \alpha_6 : A_6)$ from the original configuration formula $Conf_\gamma(\alpha_1 : A_1, \ldots, \alpha_6 : A_6)$.

Lemma A.9. *Let $\alpha_1, \ldots, \alpha_n$ and γ be nodes, $\Phi_1, \ldots, \Phi_n$ sets of formulas, F a formula such that $F \in \Phi_i$, then we have for $j \neq i$:*

$$\vdash Conf_\gamma(\ldots; \alpha_i : \Phi_i; \ldots; \alpha_j : \Phi_j; \ldots) \\ \rightarrow Conf_\gamma(\ldots; \alpha_i : \Phi_i; \ldots; \alpha_j : \Phi_j \cup \{Path_{\alpha_j,\alpha_i} F\}; \ldots)$$

Proof. By induction on the degree n of the configuration formula.

Case $n = 1$. This case is obvious.

Case $n \geq 2$. We use the definition of the configuration formula with α_j as the privileged node and get the following formula:

$$Conf_\gamma(\alpha_k : \Phi_k, k \notin J \cup \{j\}; \beta : Conf_\beta(\alpha_k : \Phi_k, k \in J) \wedge Path_{\beta,\alpha_j} \Phi_j).$$

We now proceed by case analysis, depending on whether $i \in J$ or not.

Case $i \notin J$. By induction hypothesis, we can add $Path_{\beta,\alpha_i} F$ to β and then we have $Path_{\beta,\alpha_j} \Phi_j \wedge Path_{\beta,\alpha_i} F$, where β, α_i and α_j satisfy the conditions of Lemma A.3, and we get the desired configuration formula.

Case $i \in J$. We use Lemma A.8 to add $Path_{\beta,\alpha_i} F$ to β and then use Lemma A.3 again. □

A.2. **The simulation lemmas.** We will here use our configurations to prove the result that will help us during our completeness proof.

First we will need to show that we can indeed conservatively add a configuration formula to a node, i.e. if we have the situation of Figure 2, then we can conservatively add $Conf_\gamma(\alpha_1 : A_1, \ldots, \alpha_6 : A_6)$ to γ.

Lemma A.10. *Let $\mathcal{T}$ be a tree containing nodes $\alpha_1, \ldots, \alpha_n$ and γ, and let $\Phi_1, \ldots, \Phi_n$ be finite sets of formulas such that for all i, $\Phi_i \subset \alpha_i$.*
Then we can conservatively add the formula $Conf_\gamma(\alpha_i : \Phi_i, 1 \leq i \leq n)$ to γ in $\mathcal{T}$.

Proof. By induction on the degree n of the formula.

Case $n = 1$. Adding a path formula to a node is a conservative action.

Case $n \geq 2$. By definition of the configuration formula we get:

$$Conf_\gamma(\alpha_j : \Phi_j, j \notin J \cup \{n\}; \beta : Conf_\beta(\alpha_j : \Phi_j, j \in J) \wedge Path_{\beta,\alpha_n} \Phi_n).$$

We can then add $Conf_\beta(\alpha_j : \Phi_j, j \in J)$ conservatively to β by induction hypothesis, and then $Path_{\beta,\alpha_n} \in \Phi_n$, and we finally get the result using the induction hypothesis again. □

Here is the proof of lemma 3.3 used in section 3.2.

Lemma A.11. *Let $\mathcal{T}$ be a tree, α and β two nodes and A a sentence such that $A \in \alpha$. Suppose that a finite sequence of conservative actions enables us to add B in β.*

Then there is a configuration formula we can conservatively add to α in the tree $\mathcal{T}'$, obtained from $\mathcal{T}$ by removing the formula A from the node α, so that we can then deduce in the node α of $\mathcal{T}'$ the formula $A \rightarrow Path_{\alpha,\beta} B$.

Proof. Consider our sequence of conservative actions: it involves a finite number of nodes $\alpha_1, \ldots, \alpha_n$ and some accompanying finite sets of formulas $\Phi_1, \ldots, \Phi_n$. Using Lemma A.10 we can add the needed configuration formula $G \equiv Conf_\alpha(\alpha_i : \Phi_i, 1 \leq i \leq n)$ to α, and then we can simulate the conservative actions using Lemma A.5 and Lemma A.9, because A is in α.

Using Lemma A.8, we get $\alpha \cup \{A, G\} \vdash Path_{\alpha,\beta}B$, and using Lemma A.1 we deduce $\alpha \cup \{G\} \vdash A \to Path_{\alpha,\beta}B$. □

Appendix B. Technical lemmas in the temporal transitive case

B.1. **An additional lemma.**

The proof of the generalization of axiom 5', quoted as lemma 4.2 in section 4.1.5.

Lemma B.1. *Let $\alpha_0, \ldots, \alpha_n$ be nodes, $\Phi_1, \ldots, \Phi_n$ finite sets of formulas such that $\alpha_n < \cdots < \alpha_1 < \alpha_0$ then:*

$$\begin{aligned}\vdash\ & \Diamond_P B \wedge Conf_{\alpha_0}(\alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \\ & \to \bigvee_{1 \le i \le n} Conf_{\alpha_0}(\ldots; \alpha_i : \Phi_i \cup \{B\}; \ldots) \\ \vee & \bigvee_{\substack{1 \le i \le n \\ \alpha_i < \beta < \alpha_{i-1} \\ \text{where } \beta \text{ is a new node}}} Conf_{\alpha_0}(\beta : B; \alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \\ & \vee\ Conf_{\alpha_0}(\alpha_1 : \Phi_1; \ldots; \alpha_{n-1} : \Phi_{n-1}; \alpha_n : \Phi_n \cup \{\Diamond_P B\}).\end{aligned}$$

Proof. By induction on n.

Case $n = 0$. The result is immediate.

Case $n \ge 1$. Then $Conf_{\alpha_0}(\alpha_i : \Phi_i)$ is the formula:

$$\Diamond_P((\bigwedge_{F \in \Phi_1} F) \wedge \Diamond_P((\bigwedge_{F \in \Phi_2} F) \wedge \ldots \Diamond_P(\bigwedge_{F \in \Phi_n} F) \ldots)).$$

Using axiom 5', we get:

$$\begin{aligned}&\Diamond_P B \wedge Conf_{\alpha_0}(\alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \\ &\quad \to \Diamond_P(B \wedge (\bigwedge_{F \in \Phi_1} F) \wedge \Diamond_P((\bigwedge_{F \in \Phi_2} F) \wedge \ldots \Diamond_P(\bigwedge_{F \in \Phi_n} F) \ldots)) \\ &\quad \vee \Diamond_P(B \wedge \Diamond_P((\bigwedge_{F \in \Phi_1} F) \wedge \Diamond_P((\bigwedge_{F \in \Phi_2} F) \wedge \ldots \Diamond_P(\bigwedge_{F \in \Phi_n} F) \ldots))) \\ &\quad \vee \Diamond_P((\bigwedge_{F \in \Phi_1} F) \wedge \Diamond_P B \wedge \Diamond_P((\bigwedge_{F \in \Phi_2} F) \wedge \ldots \Diamond_P(\bigwedge_{F \in \Phi_n} F) \ldots)).\end{aligned}$$

We can then use the induction hypothesis to deduce the expected configuration formula. □

Appendix C. Technical lemmas in the modal case

C.1. **Adapting configurations and other tools.**

C.1.1. *Different definitions.* As we have said in section 4.2.2, we adapt our definitions to the modal setting.

Definition C.1. Let α and β be nodes such that β is a successor of α. Then we have $\beta = \alpha b_1 \ldots b_q$. We call *path formula* from α to β for a formula B the formula $\Diamond^q B$. We write $Path_{\alpha,\beta}B$ for such a formula.

Path formulas lose some of their interest as a notation, but their use in the results is the same, so we will keep them.

Definition C.2. Let $\alpha_1, \ldots, \alpha_n$ be successor nodes of γ and let $\Phi_1, \ldots, \Phi_n$ be finite sets of formulas, we then define the notion of a *configuration formula* $Conf_\gamma(\alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n)$ by induction on n:

- if $n = 0$ by convention $Conf_\gamma(\emptyset)$ is $\top$.
- if $n = 1$ let $Conf_\gamma(\alpha_1 : \Phi_1) = Path_{\gamma,\alpha_1} \bigwedge_{F \in \Phi_1} F$.
- if $n \geq 2$, let β be the closest node to α_n on the path from γ to α_n that also belongs to a path leading from γ to an α_i; let $J \subseteq \{1, \ldots, n-1\}$ be such that $j \in J$ iff α_j is a successor of β; then let $Conf_\gamma(\alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n)$ be the formula:

$$Conf_\gamma(\alpha_j : \Phi_j, j \notin J \cup \{n\}; \beta : Path_{\beta,\alpha_n} \bigwedge_{F \in \Phi_k} F \wedge Conf_\beta(\alpha_j : \Phi_j, j \in J)).$$

C.1.2. *Different results.* We begin by proving a lemma for the propagation of $\Box$-formulas.

Lemma C.3. *Let $\alpha_1, \ldots, \alpha_n$ be successor nodes of a node γ and consider $\Phi_1, \ldots, \Phi_n$ finite sets of formulas, let A be a formula. Suppose that there is a path of length p from γ to α_k. Then we have:*

$$\vdash \Box^p A \wedge Conf_\gamma(\alpha_i : \Phi_i, 1 \leq i \leq n) \rightarrow Conf_\gamma(\ldots; \alpha_k : \Phi_k \cup \{A\}; \ldots).$$

Proof. By induction on the degree n of the configuration formula.

Case $n = 1$. This case is straightforward.

Case $n \geq 2$. Use the definition of the configuration formula and choose α_k as the privileged node. We get:

$$Conf_\gamma(\alpha_j : \Phi_j, j \notin J \cup \{k\}; \beta : Conf_\beta(\alpha_j : \Phi_j, j \in J) \wedge Path_{\beta,\alpha_k} \Phi_k).$$

Consider the path from γ to α_i: it goes through β. Suppose that the path from γ to β is of length q. By induction hypothesis, we can bring $\Box^{p-q} A$ in β, and then the result is clear. $\Box$

We now prove that we can propagate $\Box$-formulas from one node to another in a configuration.

Lemma C.4. *Let $\alpha_1, \ldots, \alpha_n$ be successor nodes of a node γ and consider $\Phi_1, \ldots, \Phi_n$ finite sets of formulas. Suppose that there is a path of length p from α_j to α_k and that $\Box^p A \in \Phi_j$. Then we have:*

$$\vdash Conf_\gamma(\alpha_i : \Phi_i, 1 \leq i \leq n) \rightarrow Conf_\gamma(\ldots; \alpha_k : \Phi_k \cup \{A\}; \ldots).$$

Proof. By induction on the degree n of the configuration formula.

Case $n = 1$. There is nothing to do.

Case $n \geq 2$. Use the definition of the configuration formula with α_j as the privileged node. There is at least α_k as a successor to α_j. Hence β is α_j. We get the formula:

$$Conf_\gamma(\alpha_i : \Phi_i, i \notin J \cup \{j\}; al_j : (\bigwedge_{F \in \Phi_j} F) \wedge Conf_{\alpha_j}(\alpha_i : \Phi_i, i \in J)).$$

Since $k \in J$ and $\Box^p A \in \Phi_j$ we can apply Lemma C.3 to get the configuration formula we needed. $\Box$

We also need to show that we can add path formulas. This was done in the temporal case using the fact that we had past and future operators and axioms expressing their symmetry. The result is still true because the path formula is in a certain sense already there in the configuration.

Lemma C.5. *Let $\alpha_1, \ldots, \alpha_n$ be successor nodes of a node γ and consider $\Phi_1, \ldots, \Phi_n$ finite sets of formulas. Suppose that α_j is a predecessor of α_k and that A is a formula such that $A \in \Phi_k$. Then we have:*

$$\vdash Conf_\gamma(\alpha_i : \Phi_i, 1 \leq i \leq n) \rightarrow Conf_\gamma(\ldots; \alpha_j : \Phi_j \cup \{Path_{\alpha_j,\alpha_k} A\}; \ldots).$$

Proof. Use the definition of the configuration formula with α_j as the privileged node. There is at least α_k as a successor to α_j. Hence β is α_j. We get the formula:

$$Conf_\gamma(\alpha_i : \Phi_i, i \notin J \cup \{j\}; \alpha_j : (\bigwedge_{F \in \Phi_j} F) \wedge Conf_{\alpha_j}(\alpha_i : \Phi_i, i \in J)).$$

Using Lemma A.8, we can deduce the formula $Path_{\alpha_j,\alpha_k} A$ from the hypothesis $Conf_\beta(\alpha_i : \Phi_i, i \in J)$, and using Lemma A.5 we can add it to α_j. □

Finally, need prove the lemma 4.6 used in section 4.2.2.

Lemma C.6. *Let $\mathcal{T}$ be a tree and α be a node in $\mathcal{T}$. Let D and $A(x)$ be formulas, where x is free in A. Suppose we can conservatively add to α $A[c/x]$ for a new constant c. Then there exists a variable w, a finite set of formulas Φ contained in α and there are nodes $\alpha_1, \ldots, \alpha_n$ in the tree, containing finite sets of formulas $\Phi_1, \ldots, \Phi_n$ such that:*

$$\vdash Conf_\epsilon(\alpha : \Phi \cup \{D\}; \alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \rightarrow Path_{\epsilon,\alpha} D \wedge \forall w A[w/x].$$

Proof. We will show a stronger result. Keep the hypothesis of the lemma. For all nodes $\gamma_1, \ldots, \gamma_m$ and formulas $C_1, \ldots, C_m$, we will show that there exists a variable w, a finite set of formulas Φ contained in α and nodes $\alpha_1, \ldots, \alpha_n$ in the tree, containing finite sets of formulas $\Phi_1, \ldots, \Phi_n$ such that:

$$\vdash Conf_\epsilon(\gamma_i : C_i, 1 \leq i \leq m; \alpha : \Phi; \alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \rightarrow Conf_\epsilon(\gamma_i : C_i, 1 \leq i \leq m; \alpha : \forall w A[w/x]).$$

By induction on the number p of conservative actions which enable us to add $A[c/x]$ in α.

Case $p = 1$. Then it must have been a deduction in α, because c is a new constant, i.e. $A[c/x]$ cannot have been propagated nor can it be a path formula. Using Lemma A.2 we get a set Φ and a variable w such that $\Phi \vdash \forall w A[w/x]$, and then $\vdash Conf_\epsilon(\gamma_i : C_i, 1 \leq i \leq m; \alpha : \Phi) \rightarrow Conf_\epsilon(\gamma_i : C_i, 1 \leq i \leq m; \alpha : \forall w A[w/x])$.

Case $p \geq 2$. If $A[c/x]$ was obtained by deduction in α, we have then a finite set Φ' such that $\vdash \bigwedge_{F[c/u] \in \Phi'} F[c/u] \rightarrow A[c/x]$, where we write $F[c/u]$ to mean that some of the F contain the constant c, i.e. are equal

to a formula $F(u)$ where we substitute c to u. If F do not contain c, just suppose that u does not appear in F. We can use Lemma A.2 to generalize this and get a variable w such that $\vdash \forall w \bigwedge_{F\in\Phi'} F[w/u] \rightarrow \forall w A[w/x]$. Then we have a shorter sequence of conservative actions which enable us to add $\bigwedge_{F\in\Phi'} F[c/u]$ in α. By induction hypothesis, we get a finite set of formulas Φ contained in α and nodes $\alpha_1, \ldots, \alpha_n$ in the tree, containing finite sets of formulas $\Phi_1, \ldots, \Phi_n$ such that:

$$\vdash Conf_\epsilon(\gamma_i : C_i, 1 \leq i \leq m; \alpha : \Phi; \alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \\ \rightarrow Conf_\epsilon(\gamma_i : C_i, 1 \leq i \leq m; \alpha : \forall w \bigwedge_{F\in\Phi'} F[w/u]).$$

We can then deduce the appropriate configuration formula.

If the last step of the sequence was to propagate a $\Box$-formula, we have a node β, such that the path from β to α is of length q, where we can add by a shorter sequence of conservative actions the formula $\Box^q A[c/x]$. By induction hypothesis, we get a variable w, a finite set of formulas Φ' contained in β and nodes $\alpha_1, \ldots, \alpha_n$ in the tree, containing finite sets of formulas $\Phi_1, \ldots, \Phi_n$ such that:

$$\vdash Conf_\epsilon(\gamma_i : C_i, 1 \leq i \leq m; \alpha : \top; \beta : \Phi'; \alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \\ \rightarrow Conf_\epsilon(\gamma_i : C_i, 1 \leq i \leq m; \alpha : \top; \beta : \forall w \Box^q A[w/x]).$$

Using the Barcan formula, we have $\vdash \forall w \Box^q A[w/x] \rightarrow \Box^q \forall w A[w/x]$, and we can then deduce the configuration formula from the above expression.

If the last step was to add a path formula, we have a node β, such that the path from α to β is of length q, where we can add by a shorter sequence of conservative actions the formula $A'[c/x]$, and $A = \Diamond^q A'$. By induction hypothesis, we get a variable w, a finite set of formulas Φ' contained in β and nodes $\alpha_1, \ldots, \alpha_n$ in the tree, containing finite sets of formulas $\Phi_1, \ldots, \Phi_n$ such that:

$$\vdash Conf_\epsilon(\gamma_i : C_i, 1 \leq i \leq m; \alpha : \top; \beta : \Phi'; \alpha_1 : \Phi_1; \ldots; \alpha_n : \Phi_n) \\ \rightarrow Conf_\epsilon(ga_i : C_i, 1 \leq i \leq m; \alpha : \top; \beta : \forall w A'[w/x]).$$

We can then add the path formula $\Diamond^q \forall A'[w/x]$ in α using Lemma C.5, and use the fact that $\vdash \Diamond^q \forall w A'[w/x] \rightarrow \forall w \Diamond^q A'[w/x]$. $\Box$

References

[1] Blackburn, P., 1993 "Nominal tense logic", *Notre Dame Journal of Formal Logic*, **34**, 56–83.

[2] Blackburn, P. and Tzakova, M., 1998a, "Hybrid completeness", *Logic Journal of the IGPL*, **6**(4), 625–650.

[3] Blackburn, P. and Tzakova, M., 1998b, "Hybridizing concept languages", *Annals of Mathematics and Artificial Intelligence*, **24**, 23–49.

[4] Bull, R., 1973, "An approach to tense logic", *Theoria*, **36**, 282–300.

[5] Gabbay, D.M., 1994, *Temporal logic vol.1: mathematical foundations*, Oxford: Oxford University Press.

[6] Gabbay, D.M., 1996, *Labelled deductive systems vol.1*, Oxford: Oxford University Press.

[7] Gabbay, D.M., 1999, *Fibring logics*. Oxford: Oxford University Press.

[8] Hughes, G.E. and Creswell, M.J., 1994, *A companion to modal logic*, London: Methuen and Co. Ltd.

[9] Passy, S. and Tinchev, T., 1991, "An essay in combinatory dynamic logic", *Information and Computation*, **93**, 263–332.
[10] Prior, A., 1967 *Past, present and future*. Oxford: Oxford University Press.

King's College, Strand, London WC2R 2LS, United Kingdom
Current address: King's College, Strand, London WC2R 2LS, United Kingdom
E-mail: dg@dcs.kcl.ac.uk

Current address: Ecole Normale Supérieure de Lyon, 46, allée d'Italie, 69364 Lyon Cedex 07, France, visiting King's College
E-mail: gmalod@ens-lyon.fr