Global View On Reactivity: Switch Graphs and Their Logics

Dov Gabbay
King's College London, UK
Bar Ilan University, Israel
University of Luxembourg, Luxembourg

Sérgio Marcelino *
SQIG - Instituto de Telecomunicações, Portugal
Departamento de Matemática, IST, Lisboa, Portugal

Abstract

The notion of reactive graph generalises the one of graph by allowing the base accessibility relation to change when its edges are traversed. Can we represent these more general structures using points and arrows? We prove this can be done by introducing higher order arrows: the switches.

The possibility of expressing the dependency of the future states of the accessibility relation on individual transitions by the use of higher-order relations, that is, coding meta-relational concepts by means of relations, strongly suggests the use of modal languages to reason directly about these structures. We introduce a hybrid modal logic for this purpose and prove its completeness.

1 Introduction

In computer science the word reactivity has been used to denote systems that react to their environment and are not meant to terminate, as coined by Pnueli and Harel in [25]. In this paper the word has a different meaning, reactive systems are history-dependent relational structures, where the accessibility relation is determined not only by the point where one is, but also by the previous transitions. This concept was introduced by Dov Gabbay in 2004, see [14] and the extended version [15]. We show that the concept of reactivity by presenting some structures that embody it and some logics to reason about them. Let us start by explaining how this concept of reactivity was born and outlining its short life-story.

*The author Sérgio Marcelino thanks the support of FCT and EU FEDER, via the postdoc scholarship SFRH/BPD/76513/2011, the PhD grant SFRH/BD/27938/2006, the project FCT PEst-OE/EEI/LA0008/2011 of IT, the FP7-PEOPLE-2012-IRSES GetFun Marie Curie International Research Staff Exchange Scheme Fellowship within the 7th European Community Framework Programme, as well as the PQDR initiative of SGIQ.

New kind of arrows. In [14], Dov Gabbay introduced the idea of enriching graph-based structures with arrows of a new type, calling it the double arrows. Double arrows, instead of connecting points, connect arrows with arrows or other double arrows, see Figure 1. The idea is that this new kind of arrows can represent the dependence of

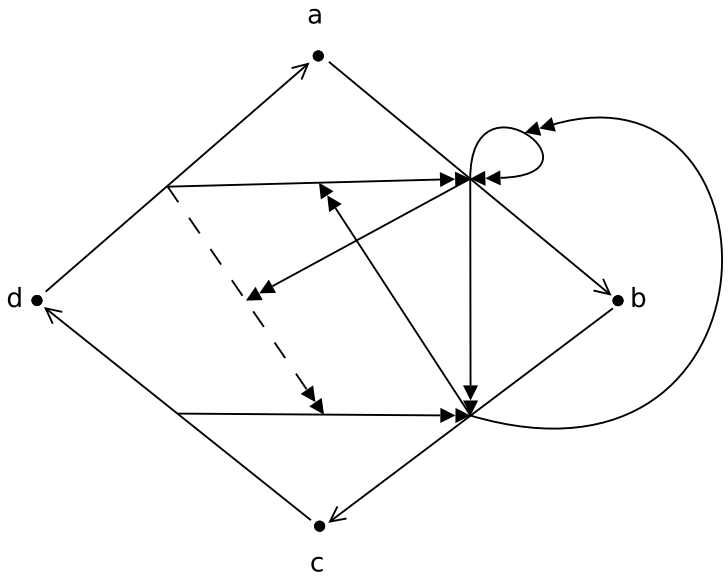

Figure 1: An enriched graph.

the state of the targeted arrow (or double arrow) upon the crossing of the arrow in its origin. In this first presentation the double arrows would simply change the targeted arrow state. Let us see how it works by playing with the example in Figure 1. We represent the fact that an arrow is off by drawing its body as a dotted line. Let us see

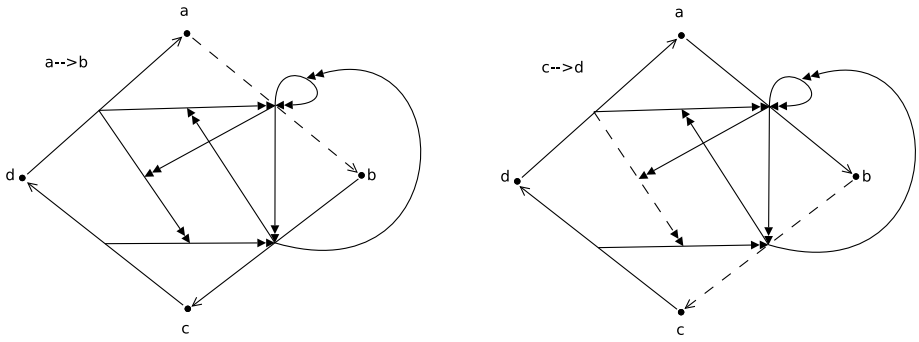

Figure 2: The effect of crossing edges.

the effect that crossing some of its edges has. As shown in Figure 2, when we cross the edges (a,b) (left) or (b,c) (right), the arrows that are in the scope of the double arrows coming out of them, become off or on if they are on or off respectively (that is, their state changes). This process is cumulative, after crossing (a,b) we can also cross (b,c) and the effects are determined by the new state of double arrows, see Figure 3. These ideas were presented using suggestive motivational cases, for example in Figure

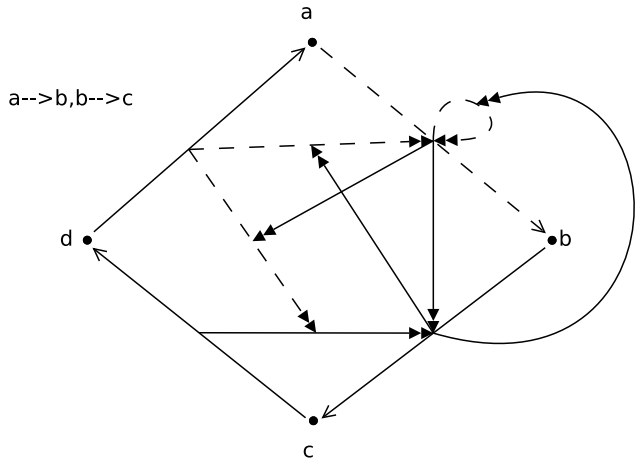

Figure 3: The effect of crossing edges.

4 we see how these new arrows can represent a classical inheritance networks case.

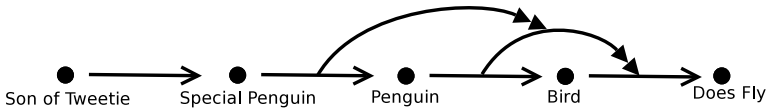

Figure 4: A classical inheritance networks example. We can represent simple exceptions: birds do fly, but although penguins are birds they do not fly. And we can also represent higher order exceptions (exceptions to exceptions): even though the son of Tweety is a special penguin (so also a penguin) he does fly.

The general idea that a relational structure may vary when one moves through it and the enriched kind of frames that came along with it fuelled publications in many areas. Indeed, there are applications of the reactive ideas in such diverse areas as modal logic, preferential non-monotonic logic, inheritance systems, context-free grammars, automata theory, deontic logic and contrary to duty, argumentation and other networks, see papers [5, 20, 12, 16, 23, 21, 22, 17, 28]. For example when one adds these kind of double arrows to the structure of an automata, one allows it to modify its transition relation while reading a sentence, that is, one makes it reactive. This alternative

paradigm competes with non-determinism in the task of obtaining automata with minimal number of states accepting a language. Indeed, the following theorem (Proposition 6.1, [12]) is proven:

> If A is deterministic automaton with k^n states, it has an equivalent reactive automaton $R(A)$ with kn states.

Another interesting application of this kind of enriched graphs (though not using the dynamical counterpart) can be found in [28], where Dung's abstract argumentation theory is extended incorporating the meta-level argumentation-based reasoning, about possibly conflicting preferences between arguments.

Changing Kripke structures One can say that the relational semantics of modal logic already encompasses change. In fact, one can consider (or access to) different worlds, and the propositional truths, given by the propositional variables valuation, may change. Also the accessible worlds may change with these transitions. Yet, the truths at a given world are 'still'. In a Kripke model both propositional truths and the accessible worlds are fixed for each world. One can take it a step further and let these vary. In many situations it makes sense to consider semantics such that when certain operators are evaluated, the model, where the formula is being evaluated, changes. Therefore, the interpretation of a formula in the scope of a modal operator is given by a general condition of the type:

$$\mathfrak{M}, x \vDash \Diamond\varphi \text{ if } \mathfrak{M}', x' \vDash \varphi$$

where x' is a point in a new model $\mathfrak{M}'$. Indeed there are various examples of such approaches, e.g.:

- in dynamic epistemic logics with agent's public announcements [33];
- in sabotage logics edges can be deleted [32];
- in memory logics one may keep the information that a certain world was visited, adding it to the memory of the model [4];
- in Hyper-modalities the meaning of the modal operators depends on where in the formula they occur [13];
- in product logics one may think that while moving along one direction the valuation of the remaining hyper-plan is changing, for example modelling valuation change in time if that direction is a time flow [18].

Local view: reactive logics, making Kripke reactive In [15] Kripke structures are made reactive. Gabbay introduces a semantics based on Kripke frames enriched with double arrows, where the basic relation changes along the interpretation of a formula by the action of the double arrows. The dependence is on where one has been before, that is, neither relation nor propositional variables values change with the clock ticking but they react when and because we move. Moreover, the changes are sensitive to the

way we got to the current world. Subsequently it is proven that this semantics strictly generalises the Kripke semantics, in fact, we may have classes of these frames originating logics that are not closed under substitution. It is not claimed that the classical Kripke frames cannot cope with these types of change, the matter is more about how to incorporate these meta-level notions into the models and which language to consider in order to reason about them. In [19, 27] is introduced a more abstract notion of reactive Kripke frames. Whereas in [15] the changes in the accessible relation are the ones produced by the action of the double arrows, in the abstract notion of reactive Kripke frames these changes are given. In reality, in the usual semantics of modal logic the only important information to the value of a modal formula is the set of successors at each moment, i.e the **local** accessibility relation. Therefore the notion of reactive Kripke frame boils down to a set of admissible sequences of points, that is, the set of admissible paths. One can picture the initial accessibility relation by considering the paths of size two, and its evolution is encoded in the one step prolongment relation on the bigger paths. The semantics presented in [15] is generalised over these abstract structures, allowing the valuation to vary along the paths, and the language is enriched with an extra operator relating paths that have the same endpoint. This is similar to what it is done in the branching-time logic with quantification over branches in [34]. Finally results of soundness and completeness are presented characterising some properties of these frames (generalising some familiar properties of 'static' Kripke frames: reflexivity, symmetry and transitivity). Furthermore some results regarding the decidability of the resulting logics are obtained. Let us look to a concrete case and see some examples of what can be expressed in the considered language.

Example 1.1. Let us consider the situation of a traveller with a budget. The set of his possible moves depends on whether he has enough money to do them (to pay tolls, oil, train or flight tickets), furthermore his actual moves also determine his future possibilities. So the paths of the correspondent reactive frame are the sequences of cities he can visit with a certain budget. The formulas are interpreted over these paths. Let $\Diamond_R$ stand for the dynamics operator, that is, corresponding to the accessibility relation, and $\Diamond_P$ to the relation identifying the paths with the same endpoint. So, $\Diamond_R\varphi$ means that after the current path we can access to a city such that the resulting path satisfies φ. $\Diamond_R\varphi$ means that there is a path to the current world satisfying φ. Let us consider that the propositional symbols p_b and p_w correspond to the predicate of being able to buy bread and wine respectively, and m be true if there is still some money left. See Table 1 for examples of what can be said.

Modal language	**Natural language**
$\neg m \to \Box_R \bot$	If the traveller has no money left then he cannot move
$\Diamond_P(\Box_R p_b \wedge \Diamond_R \top)$	There is a path to the current city, after which the traveller is not blocked and he has enough money to buy bread every city he can access to
$(p_w \wedge \Box_R p_w) \to \Box_P p_b$	If the traveller can buy wine now, and at any immediate next stop, then he would always be able to buy bread in the current city regardless of how he got there

Table 1: Possible statements in the considered modal semantics.

The truth values of the sentences in Table 1 depend on the particular valuation we pick, but we noticed above, with this language we can capture interesting structural aspects of these frames (see [19]).

1.1 Global view: Switch graphs and their frames

While the notion of reactive frame contains exactly the necessary information to generalise the usual Kripke semantics to the reactive case, it ignores the state of the global accessibility relation. In order to model this global dependence we consider the concept of reactive graphs. Intuitively a reactive graph consists in a graph that may change its configuration when a certain edge is crossed.

As in the case of reactive frames, we define a reactive graph to be a set of admissible sequences of edges. Clearly, from such a set one can extract the evolution of the whole relational structure while transversing the graph edges. For any admissible sequence λ, the relational state of the reactive graph after λ is given by

$$R_\lambda = \{(w, w') : \lambda(w, w') \text{ is an admissible sequence of edges}\}.$$

At this point a natural question arises: can all these relational behaviours be encoded by double arrows? In order to answer this question we introduce the concept of a switch graph.

Switch graphs A switch graph is a graph enriched with two kinds of double arrows, the connecting and the disconnecting switches. As their names suggest, when the origin of a connecting/disconnecting switch is crossed its target is connected/disconnected.

Given a set W, a switch is either an edge $s \in W^2$ (switch of level 0, neither connecting or disconnecting) or a triple $s = (a, s', *)$ (of level $n > 0$), where

- $a \in W^2$ in the edge that triggers its action,
- s' is the targeted switch (of level $n - 1$)
- $* \in \{\bullet, \circ\}$ says if it is a connecting (black circle) or disconnecting (white circle) switch.

The type of the switches of level 0 is ϵ (the empty sequence) and of $s = (a, s', *)$ is $\sigma *$ where σ is the type of s'. We use the following notation to refer to switches in an easier fashion:

- $(ab, \epsilon) = (a, b)$,
- $(v_1v_2, \ldots, v_{2n+1}v_{2n+2}, a, *_1 \ldots *_{n+1}) = ((v_1, v_2), (v_3v_4, \ldots, v_{2n+1}v_{2n+2}, *_1 \ldots *_n), *_{n+1})$.

In graphical representations we use white headed arrows to represent the disconnecting switches and black headed arrow to represent the connecting ones. Let us see an example of a situation where the dynamical restrictions are easily represented by these structures.

Example 1.2. Switches can easily grasp the fact that certain resources are finite, that is, one can use them a finite number of times. Depending on the meaning of the accessibility relation (e.g. crossing a bridge, driving a road, taking a pill from a tablet, printing pages, ask a person for a cigarette, etc) the switch configuration presented in Figure 1.2 represents the fact that a particular action can be taken exactly $k > 1$ number of times. The set of switches is given by $\{(a,b),(ab,\ldots,ab,\circ\bullet^{k-1})\}$. For $k = 0$ we would have (a,b) and $(ab,ab,\circ)$ on, and would not need more switches.

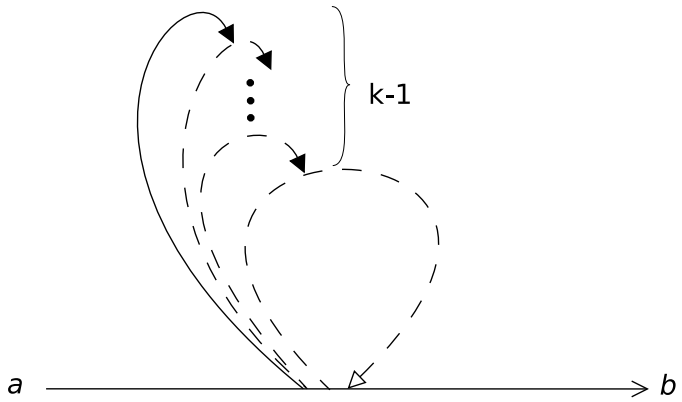

Figure 5: The edge (a,b) can be crossed exactly $k > 1$ times.

The main result of section 2 answers to the above question, we prove that any reactive graph can be generated by a switch graph. We also provide an example where switches are used to represent the dynamical restrictions required for the solution of the mutual exclusion problem.

Switch reactive hybrid logic In order to reason about switches and their actions we introduce an interpretation of a hybrid modal language over Kripke frames generated by switch graphs dynamics. Usually nominals are valid in exactly one point in each model. In the reactive setting they are valid in exactly one point for each of the components representing different reactive moments, with different relational states. The general idea is has follows:

- Instead of having an operator relating the different relational states of a point, we use nominals to identify them.
- Also, using the fact that double arrows are relations (of different arities) over the carrier, we consider the correspondent modal operators. To each type of switches we consider the associated $2n+2$ relation

  $$R^{\sigma} = \{(w_1, w_2, \ldots, w_{2n+1}w_{2n+2}) : (w_1w_2, \ldots, w_{2n+1}w_{2n+1}, \sigma) \textbf{ is on}.\}$$

  for $|\sigma| = n$, and its correspondent modal operator $2n+1$-ary modal operator $\Diamond_{\sigma}$.

- Moreover the use of the hybrid operator @ allows us to have a global view over the switch configuration at each moment.
- To the fragment of the language introduced above, that allows us to talk about the switches state in each moment, we add the modal operator $\diamondsuit$ relating the different states of the switch graph, being the real dynamics operator (corresponding to $\diamondsuit_R$ in the reactive Kripke frames).

In section 3 we give an axiomatisation of the switches evolution by capturing the interaction between these components but for now let us look at some examples of what we may express in this language.

Example 1.3. Let us consider a non-local version of Example 1.1. Instead of considering a single traveller, that can be only at one place at a time, let us consider the same problem but with a truck company (or group of travellers) with a common budget. Clearly each move affects all the subsequent possible moves. We know from the result proven in Chapter 2 that any such reactive dynamics can be expressed by switches. If we consider a switch graph generating this dynamics and the language described above we may express the local interdependencies explicitly, e.g.

$$@_a \wedge \Diamond\!\!\!\!\cdot_{\circ}(b, c, d)$$

means that the fact that a truck in a goes to b implies that no truck in c can go to d. That is

$$@_a \diamondsuit (b \to @_c \neg \diamondsuit d).$$

In the case of Example 1.2 dealing with bounded resources, where we allow only certain kinds of switches:

$$@_a(\Diamond\!\!\!\!\cdot_{\epsilon} b \wedge \Diamond\!\!\!\!\cdot_{\circ}(b, a, b))$$

means that (a, b) can be crossed exactly once, $@_a \diamondsuit (b \to @_a \neg \diamondsuit b)$,

$$@_a(\Diamond\!\!\!\!\cdot_{\epsilon} b \wedge \neg \Diamond\!\!\!\!\cdot_{\circ} (b, a, b) \wedge \neg \Diamond\!\!\!\!\cdot_{\circ\bullet} (b, a, b, a, b) \wedge \Diamond\!\!\!\!\cdot_{\circ\bullet\bullet}(b, a, b, a, b, a, b))$$

means that (a, b) can be crossed exactly three times, $@_a \diamondsuit (b \to @_a \diamondsuit (b \to @_a \neg \diamondsuit b))$, and so on.

We do not claim that this is the most appropriate language to reason about all cases of reactivity. For instance, in Examples 1.1 and 1.3 we can envisage a language that could explicitly reason about the cost of each move and the remaining budget after it. And in the case of Example 1.2, where only some shapes of switches are allowed, the language could be simplified. Still, the fact is that all reactive systems can be generated by switches and this language seems adequate to express the local dynamic dependencies on each move, imposed by the switches. We hope that it represents a kind of skeleton for the various possibilities. In the conclusion of section 3 we discuss possible extensions of this language with operators of the kind we find in CTL and CTL^*. The main result of this section is the proof of the usual hybrid completeness result in this dynamic context. An important fact to retain is that in each moment the whole future relational dynamics is coded in the switches, so it may be that some properties (that depend only on the worlds - corresponding to pure formulas, having no propositional symbol that is not a nominal) can be derived by reasoning locally, using only the information contained in switches configuration.

2 Switch graphs

A graph is the abstract representation of a binary relation between objects. There are many graph-based structures, and often, to read (some of) the information represented in them, one needs to travel across their vertices following their edges, e.g. to check if two points are connected or to interpret modal formulas in Kripke model. In the usual notion of graph the vertices that are accessible from a vertex are fixed.

When we modify the notion of graph allowing that the accessible vertices depend on the sequence of edges we have crossed we obtain a reactive graph. So, to each sequence of crossed edges corresponds a (relational) state of the graph, where the edges that are available are the ones that can prolong the current sequence. The information in such a graph boils down to the sequences of edges (or actions) over a certain set.

A direct way of representing such a graph would be to draw the tree given by the admissible sequences of edges, and, perhaps, to draw at each of its nodes the state of the graph at that point. Although this idea of having different points representing the relational state of the same point is useful (in [19] we used it to obtain a relation with classical modal logic), it does not offer an easy reading of the changes that crossing an edge implies. As stated in the introduction, the concept of reactivity was initially introduced using an enrichment of graphs with new kinds of arrows representing the local effects of traversing an edge in the global relation. The representation offered by these structures seems to be much more interesting as it truly grasps the reactivity flavour. Furthermore, we prove that these new multi-level-arrowed structures are enough to code all the relational dynamics (Theorem 2.8).

The simplest effect crossing an edge can have over the accessibility relation is to turn on or off a connection. These elementary changes can be represented by drawing an arrow from the crossed edge to the edge representing the connection being altered. Crossing an edge does not have necessarily always the same effects. To represent these changes we use arrows from the edges to the arrows representing the elementary effects, and so on, obtaining infinite levels of arrows.

In this context we refer to the arrows as switches. The edges connecting points are 0-level switches and the switches connecting 0-level and n-level switches are $n+1$-level switches. The switches of level greater than 0 can be of two kinds, the connecting and disconnecting ones. A set equipped with a set of switches is called switch graph.

The state of the relation after a certain sequence of actions in a switch graph takes into account its levelled structure. The switches of level bigger than 0 do not represent the accessibility relation between points but how this relation changes after each action, corresponding to the 0-level arrows. When we cross an edge we turn on/off the switches that are the targets of the connecting/disconnecting switches coming out of it.

In Figure 6 we can see the switch graph representing the reactive graph with set of points $\{a,b\}$, and admitting as set of actions the set given by the following regular expression (being $\{a,b\}^2$ the alphabet):

$$((a,a)+(a,b))^*(b,b)((a,b)+(b,b))^*(a,a)((a,a)+(b,b))^*.$$

We see that once (b,b) is crossed, the disconnecting switch coming from (a,a) to

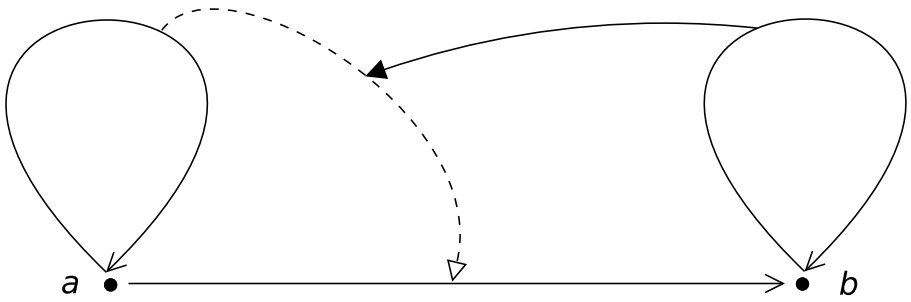

Figure 6: The white pointed arrows represent the connecting switches, the black arrows the disconnecting ones and the dashed line the fact that the switch is initially disconnected.

(a, b) becomes connected. Thus, after crossing (b, b) and (a, a), (a, b) can no longer be crossed.

Many other kinds of switches, having different effects, could be considered. These seem to be the most primitive ones, since they incorporate the basic actions in graph reactivity, connecting and disconnecting edges. We shall see that this is enough to represent all reactive graphs. That is: any reactive behaviour can be decomposed into this kind of local actions.

2.1 Reactive by Switch

Definition 2.1. A *reactive graph* is a pair (W, Δ), where

- W is a non-empty set, the set of *worlds*, and
- Δ, the *behaviour*, is the set of admissible sequences of edges, i.e., a subset of $(W \times W)^*$ closed under prefixes containing ϵ (the empty sequence is always admissible).

The state of the accessible relation after an admissible sequence of edges λ being covered is given by

$$R_\lambda = \{(w, w') : \lambda(w, w') \in \Delta\}.$$

We now introduce the enriched notion of graph, the switch graphs, and formalise in which sense they can be used to represent (or generate) reactive graphs highlighting the effects that moving around the base graph has on the accessibility relation.

Definition 2.2. For a non-empty set W and $n < \omega$, the set $\mathcal{A}_n(W)$ of *switches* over W of level n is defined as:

- $\mathcal{A}_0(W) = W \times W$ (of level 0),
- $\mathcal{A}_{n+1}(W) = (W \times W) \times \mathcal{A}_n(W) \times \{\bullet, \circ\}$ (of level n).

A switch with $\bullet$ ($\circ$) as its third component is called a *connecting (disconnecting) switch*.

The set of all switches on W is defined by taking $\mathcal{A}(W) = \bigcup_{n<\omega} \mathcal{A}_n(W)$.

Definition 2.3. A *switch graph* is a pair (W, R), where

- W is a non-empty set, the set of *worlds*, and
- $R \subseteq \mathcal{A}(W)$ is the set of *switches*.

We say that (W, R) has *reactivity of level n* if $R \subseteq \bigcup_{i\leq n} \mathcal{A}_i(W)$.

Remark 2.4. In the graphical representation we use normal arrows for the 0-level switches, a black (white) pointed arrow for the connecting (disconnecting) switches and switches that are off are drawn with a dashed line.

Here we introduce some notation to refer to switches in an easier fashion:

Definition 2.5. $(v_1v_2, \ldots, v_{2n-1}v_{2n}, a, \sigma)$ for $\sigma = s_1 \ldots s_n \in \{\bullet, \circ\}^n$, $n < \omega$ and $a \in \mathcal{A}(W)$ is defined as:

- $(a, \epsilon) = a$,
- $(v_1v_2, \ldots, v_{2n+1}v_{2n+2}, a, s_1 \ldots s_{n+1}) = ((v_1, v_2), (v_3v_4, \ldots, v_{2n+1}v_{2n+2}, a, s_1 \ldots s_n), s_{n+1})$.

We say that $(v_1v_2, \ldots, v_{2n-1}v_{2n}, \sigma)$ is a switch of type σ.
For example,

$$(w_1w_2, w_3w_4, w_5, w_6, \circ \bullet \bullet)_p = \Big((w_1, w_2), \big((w_2, w_3), ((w_3, w_4), (w_4, w_5), \circ), \bullet\big), \bullet\Big)$$

is of type $\circ \bullet \bullet$.

Definition 2.6. Given a switch graph $S = (W, R)$, the *behaviour of* S is the smallest set, Δ_S, such that:

- $\epsilon \in \Delta_S$,
- If $\alpha \in \Delta_S^n$ then $\alpha(w, w') \in \Delta_S$ for all $(w, w') \in R_\alpha$.

  Where $R_\alpha \subseteq \mathcal{A}(W)$ is the *switch state* of our switch graph after crossing the sequence of edges α:

  - $R_\epsilon = R$, the initial state,
  - $R_{\alpha(w,w')} = (R_\alpha - \{a : ((w, w'), a, \circ) \in R_\alpha\}) \cup \{a : ((w, w'), a, \bullet) \in R_\alpha\}$.

(W, Δ_S) is the reactive graph generated by S and

$$\Delta_S = \{R_\alpha : \alpha \in \Delta_S\}$$

of *switch states* of S.

Notice that the reactive level of R_α is the same for all α. Neither the highest order arrows can ever be turned off, nor higher order ones can be introduced, since a switch that acts over a switch of order n has order $n + 1$.

Remark 2.7. When we informally introduced the switches dynamics we did not specify what should happen when a connecting and a disconnecting switch act simultaneously over the same switch. In the previous definition the convention is that the connecting action prevails. We could have opted instead for:

- the disconnecting switch would prevail

  $R_{\alpha(w,w')} = (R_\alpha \cup \{a : ((w,w'),a,\bullet) \in R_\alpha\}) - \{a : ((w,w'),a,\circ) \in R_\alpha\}$,

- or that it would depend on the state of the target, by, for example, always changing its state

  $R_{\alpha(w,w')} = (R_\alpha - \{a : ((w,w'),a,\circ) \in R_\alpha\}) \cup \{a : ((w,w'),a,\bullet) \in R_\alpha \ \&\ a \notin R_\alpha\}$.

We are interested in studying how expressive these structures are regarding the generation of reactive graphs. It is easy to see that given a reactive graph $\mathfrak{F} = (W,\Delta)$ the switch graph $\mathfrak{G} = (W, R^\circ \cup R^\bullet)$ where

$$\begin{aligned} R^\bullet &= \{(w_1w_2,\ldots,w_{2|\sigma|+1}w_{2|\sigma|+2},\sigma) : \sigma \in \{\bullet\}^*,\ w_1\ldots w_{2|\sigma|+2} \in \Delta\},\\ R^\circ &= \{(vv',w_1w_2,\ldots,w_{2|\sigma|+1}w_{2|\sigma|+2},\sigma\circ) : \sigma \in \{\bullet\}^*,\ v,v',w_i \in W\}, \end{aligned}$$

generates $\mathfrak{F}$ *if the connecting switches prevail*. For the other options there does not seem to be such a direct way of coding the reactive behaviour. We can then ask whether these options are relevant for this goal. Next theorem shows that if one allows unbounded levels, these options do not limit the switch graphs expressivity. Furthermore, it presents a general construction such that given any reactive graph we obtain a switch graph that, regardless of the chosen option, generates the given reactive graph.

Theorem 2.8. *Any reactive graph can be generated by a switch graph, furthermore this switch graph can be chosen such that no connecting and disconnecting switches ever act simultaneously over the same switch.*

Proof. Given a reactive graph (W,Δ), we define a relation $C \subseteq \{\bullet,\circ\}^* \times (W \times W)^*$ by taking

$$C(\sigma,\alpha) \quad \text{iff} \quad \begin{array}{l}\text{either the number of } \circ\text{s in } \sigma \text{ is even and } \alpha \in \Delta,\\ \text{or the number of } \circ\text{s in } \sigma \text{ is odd and } \alpha \notin \Delta.\end{array}$$

This definition clearly implies the following:

Lemma 2.9. $C(\sigma\bullet,\alpha) \leftrightarrow \neg C(\sigma\circ,\alpha)$.

Now let

$$R = \{(w_1w_2,\ldots,w_{2|\sigma|+1}w_{2|\sigma|+2},\sigma) : C(\sigma,(w_1,w_2)\ldots(w_{2|\sigma|+1},w_{2|\sigma|+2})),\ \sigma \in \{\bullet,\circ\}^*,\ w_i \in W \text{ and } w_1\ldots w_{2|\sigma|} \in \Delta\}.$$

We claim that, for every $\alpha \in \Delta_S$, the following hold:

Lemma 2.10. *For every* $(\beta,\sigma) \in \mathcal{A}(W)$, $(\beta,\sigma) \in R_\alpha \iff C(\sigma,\alpha\beta)$.

We prove the lemma by induction on α. For $\alpha = \epsilon$ this is just the definition of R. Now suppose that $\alpha(w,w') \in \Delta_S$ and Lemma 2.10 holds for α. Then, by Lemma 2.9, we have that

$$((w,w')\beta, \sigma\bullet) \in R_\alpha \iff ((w,w')\beta, \sigma\circ) \notin R_\alpha. \quad (1)$$

Now by (1) we have:

$$\text{for every } (\beta,\sigma) \in \mathcal{A}(W), \quad (\beta,\sigma) \in R_{\alpha(w,w')} \iff ((w,w')\beta,\sigma\bullet) \in R_\alpha. \quad (2)$$

Now we can show Lemma 2.10 for $\alpha(w,w')$:

$$(\beta,\sigma) \in R_{\alpha(w,w')}$$

iff (by (2))

$$((w,w')\beta,\sigma\bullet) \in R_\alpha$$

iff (by the IH)

$$C(\sigma\bullet, \alpha(w,w')\beta)$$

iff

$$C(\sigma, \alpha(w,w')\beta).$$

Now we can complete the proof of the theorem as follows. First, $\epsilon \in \Delta \cap \Delta_S$. Otherwise, $\alpha(w,w') \in \Delta_S$ iff $\alpha \in \Delta_S$ and $(w,w') \in R_\alpha$ iff (by Lemma 2.10) $\alpha \in \Delta_S$ and $C(\epsilon, \alpha(w,w'))$ iff $\alpha \in \Delta_S$ and $\alpha(w,w') \in \Delta$ iff $\alpha(w,w') \in \Delta$. □

Remark 2.11. *Notice that the condition that $w_1 \dots w_{2|\sigma|} \in \Delta$ in the definition of R is not necessary but avoids the inclusion of switches that will have no part in the dynamics. Indeed it is easy to see that in what respects to the behaviour of a switch graph only such switches matter.*

2.2 Modelling multiple agents or processes

Reactive/Switch graphs can model situations where the accessibility relations change when an edge is crossed. Without any limitation on the number of individuals going through its edges. In fact a reactive graph may be used to design a particular interaction between many agents going around in a graph.

In a k-agents (processes, individuals, etc.) setting, the switch configuration ceases to be the only relevant information, we need to keep track of each agent's position. The k-behaviour of a switch graph with set of initial configurations $C \subseteq W^k$ is the set of allowed sequences of moves when the k agents are located they start in positions in C elements are allowed to do. Let us formalise this notion.

Definition 2.12. *Given a reactive graph $\mathcal{R} = (W,\Delta)$ and a number k of agents wandering about in the graph we define:*

- *$(W^k)^{\%}$ is the subset of $(W^k)^+$ formed by the sequences where each element differs from its successor of only one component, that is:*
  - $W^k \in (W^k)^{\%}$,

– $\alpha(w_1,\ldots,w_i,\ldots,w_k) \in (W^k)^{\%}$ *then* $\alpha(w_1,\ldots,w_i,\ldots,w_k)[i \to w] \in (W^k)^{\%}$ *for any* $1 \leq i \leq n$, *where*

$$\alpha(w_1,\ldots,w_i,\ldots,w_k)[i \to w] = \alpha(w_1,\ldots,w_i,\ldots,w_k)(w_1,\ldots,w,\ldots,w_k).$$

- $\mathcal{E} : (W^k)^{\%} \to (W \times W)^+$ *is defined as:*

  – $\alpha \in W^k$ *then* $\mathcal{E}(\alpha) = \epsilon$,

  – $\alpha = \alpha'(w_1,\ldots,w_i,\ldots,w_k)[i \to w] \in (W^k)^{\%}$ *then*

$$\mathcal{E}(\alpha) = \mathcal{E}(\alpha(w_1,\ldots,w_i,\ldots,w_k))(w_i,w).$$

Let $C \subseteq W^k$ *be a set of initial allowed configurations for the k-agents, C generated k-behaviour is:*

$$\Delta_C^k = \{\alpha \in (W^k)^{\%} : \mathcal{E}(\alpha) \in \Delta^k \text{ and exists } \gamma \in C \text{ that is a prefix of } \alpha\}.$$

Switch graphs can be useful in modelling programs or protocols, capturing the intended interaction between the entities involved. The design of the constraints can be directly imposed by strategically locating the appropriate switches. Ideally, this double perspective on the dynamics, would allow that the verification of such properties over a switch graph S, departing from a given configuration γ, could either be extracted from the shape of S switches or exhaustively checked over Δ_C^k.

Example 2.13. Let us consider the mutual exclusion problem, taken from [26]. There we can find a model-based approach to the verification of the required properties in a given system, presented as the solution to this problem. And, the idea is to code the intended properties in CTL and verify if the transition system associated to the solution satisfies them. Here we lay the basis for a different approach.

As we referred in the introduction, in [12] by adding higher order arrows to the structure of an automata, and thus allowing its transition table to change while it is reading a sentence, the authors achieved an exponential reduction in the minimal number of states needed to accept a language. Therefore, this way of representing systems may have impact in model checking, where the state explosion problem is a serious drawback. The switches configuration at each point determines all the future dynamics, coding the interdependence of actions. Of course that if one obtains less states when considering the switch graph corresponding to a certain protocol instead of considering its associated transition system, is not because some information was thrown away. The fact is that this information is coded in a different form, hopefully in a more intuitive and accessible way. It is the extra expressivity, given by the higher order arrows, that allows us to identify different states with the same point by associating them to different switches configuration. Moreover, if one finds an appropriate way to extract the corresponding switch graph of a program or protocol, one may expect that the verification of some properties can be reduced (by means of some intermediary reasoning) to a simpler verification over the switches. Although we do not develop the verification part, in the next section we introduce a language to reason about switches and their effects. We believe that this language can be extended to a point where we

can reason about both the switch view and the usual transition system view, and derive enough knowledge about their interaction in such a way that in order to verify some properties at one level it is enough to guarantee some related properties in the other and vice-versa. Nevertheless, here, we concentrate on showing how switches can be integrated in transition systems to improve their modelling expressivity. We understand that in this case the switches are associated to conditional commands and changes on the auxiliary variables used in these conditions. However, no systematic knowledge about that connection was obtained yet. We will limit ourselves to present examples of switch graphs 'associated' (in an informal and intuitive way) to protocols that constitute possible solutions to the problem and discuss them.

The mutual exclusion problem as presented in Chapter 3 of [26] is:

> *When concurrent processes share a resource (such as a file on a disk or a database entry), it may be necessary to ensure that they do not have access to it at the same time. Several processes simultaneously editing the same file would not be desirable. We therefore identify certain critical sections of each process's code and arrange that only one process can be in its critical section at a time. The critical section should include all the access to the shared resource (though it should be as small as possible so that no unnecessary exclusion takes place). The problem we are faced with is to find a protocol for determining which process is allowed to enter its critical section at which time. Once we have found one which we think it works, we verify our solution by checking that it has some expected properties, such as the following ones:*
>
> ***Safety**: The protocol allows only one process to be in its critical section at any time.*
>
> *This safety property is not enough, since a protocol which permanently excludes every process from its critical section would be safe, but not very useful. Therefore, we should also require:*
>
> ***Liveness**: Whenever any process wants to enter its critical section, it will eventually be permitted to do so.*
>
> ***Non-blocking:** A process can always request to enter its critical section.*
>
> *Some rather crude protocols might work on the basis that they cycle through the processes, making each one in turn enter its critical section. Since it might be naturally the case that some of them request accesses to the shared resource more than others, we should make sure our protocol has the property:*
>
> ***No strict sequencing:** Processes need not enter their critical section in strict sequence.*

We model N processes, each of which is in its non-critical state (n), or trying to enter its critical state (t), or in its critical state (c). Each individual process undergoes transitions in the cycle $n \to t \to c \to n \to ...$, see in Figure 7[1] the case $N = 2$, but the two processes interleave with each other.

[1] In the graphical representation one finds numbers juxtaposed to the letters, this is due a limitation of the

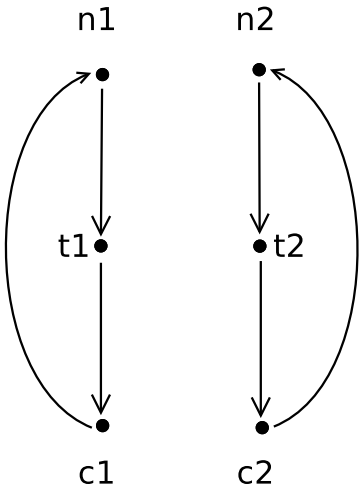

Figure 7: The mutual exclusion problem: combining processes sharing resources, two processes scenario.

One immediate solution (for $N = 2$) one can think of, is to use switches to impose directly the condition that one process gets to its critical area the other cannot – using $(t_1c_1, t_2c_2, \circ)$ and $(t_2c_2, t_1c_1, \circ)$ – and to remove this restriction when it returns to the non-critical state – using $(c_1n_1, t_2c_2, \bullet)$ and $(c_2n_2, t_1c_1, \circ)$, see Figure 8. The two processes start off in their non-critical states as indicated by the incoming edges with no source. Either of them may now move to its trying state, but only one of them can ever make a transition at a time (asynchronous interleaving). The problem here is that nothing prevents one of the processes from getting stuck at the trying state while the other one accesses continuously to the critical area, so liveness fails.

A way to guarantee liveness is, when a process requires permission to move to the critical state (moves to state t), to allow the other process to require access the critical area only once until the first process accesses to it. In Figure 9 we can see that the extra switches do exactly this. The situation is symmetric so lets check liveness for the first process, that is, if it moves to t_1, it will eventually be able to move to c_1. If process 1 moves to t_1, it turns on $(c_2n_2, n_2t_2, \circ)$ (by the action of $(n_1t_1, c_2n_2, n_2t_2, \circ\bullet)$), which guarantees that if process 2 does the whole cycle once, then (since $(c_2n_2, n_2t_2, \circ)$ is on) it can only try again if process 2 passes by c_2, $(t_1c_1, n_2t_2, \bullet)$ and $(t_1c_1, c_2n_2, n_2t_2, \circ\circ)$ remove that restriction. But this is done by restricting process's 2 ability to *require* access to its critical section, thus failing the non-blocking constraint.

This is easily solved in Figure 10, where the limitation is forced only at the last stage. One can see that the two switch graphs are really similar, the difference is that the

application used, thus they should be seen has being underscripts. Outside the figures we will use the proper form to avoid that with their intense use the text becomes unreadable. E.g. $t1$ in the figure corresponds to t_1 in the text.

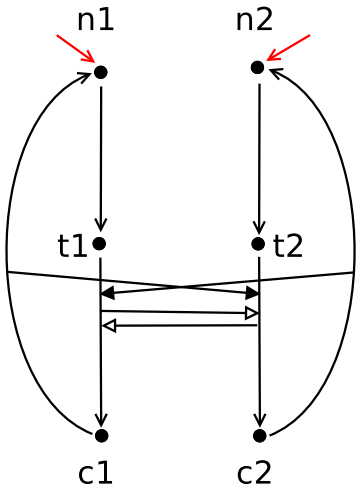

Figure 8: $N = 2$: liveness fails.

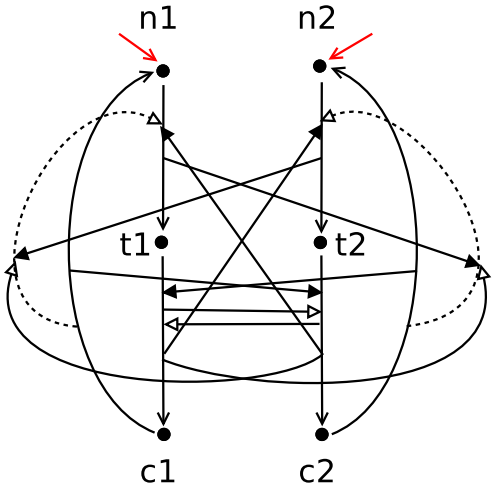

Figure 9: $N = 2$: non-blocking fails.

disconnected edge is (t_2, c_2) instead of (n_2, t_2). That is, instead of using the following set of switches

$$\{(n_i t_i, c_j n_j, n_j t_j, \circ\bullet), (t_i c_i, n_j t_j, \bullet), (t_i c_i, c_j n_j, n_j t_j, \circ\circ) : i, j \in \{1, 2\}, i \neq j\},$$

we use

$$\{(n_i t_i, c_j n_j, t_j c_j, \circ\bullet), (t_i c_i, t_j c_j, \bullet), (t_i c_i, c_j n_j, t_j c_j, \circ\circ) : i, j \in \{1, 2\}, i \neq j\}.$$

So when one process requires to access to the critical state, the other may access only once to it, though being still able require access to it, until the first's access is granted (other levels could be as easily set). Thus this solution complies with all the problem requirements.

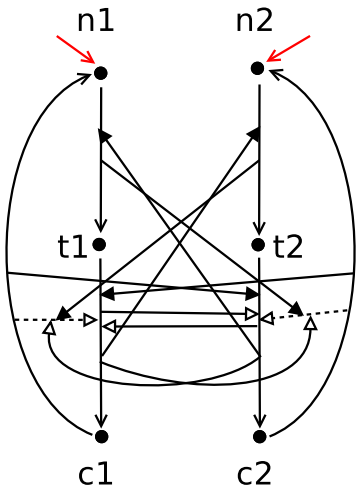

Figure 10: $N = 2$: a complete solution.

Hence, the advantage of the switch graph representation is that we can force/check the properties at the meta-level. We directly observe the effect that each transition has on the global accessibility relation. This strongly suggests how these solutions could be implemented by programming. Though the other direction (from programs to switch graphs) is not approached here, it seems a fundamental step in evaluating these structures potential.

Till now we tried solutions where each process runs around on different connected components of the presented switch graph, but there is no reason to be so. If we allow more process to run around the solutions in 9 and 10, then both liveliness and non-blocking properties would clearly be lost. Let us look to some solutions with only one connected component where all process run, thus saving in the number of required points in the graph.

In Figure 11 we have a general straightforward solution for the N processes case, but again liveness is not guaranteed. Again nothing forbids a process (or a group of $N' < N$ processes) to keep accessing the critical area making it impossible for some processes to do it. There does not seem to be a way of avoiding this with this base graph (without losing the Non-blocking property), at least if we do not use multiple connection between each point, which are not allowed in our definition of switch graphs. Of course it would be easy to consider labelled switch graphs where this was allowed but it would be closer to considering different connected components.

Instead, in order to guarantee the other properties we will consider a slightly more

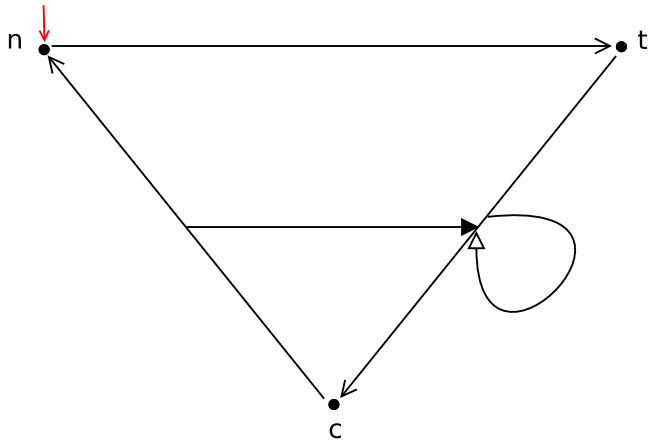

Figure 11: N arbitrary: safety and no strict sequence only.

complex base graph. We use a number of 'trying' states equal to the number of processes. We learned the kind of restrictions needed from Figure 10 and we can see in Figure 12 a switch graph using the same idea for avoiding the loss of liveness. Since here both processes share the same graph point for non-critical and critical sections the way of imposing it changes slightly. First we need to guarantee we do not have more than one process in the same point representing 'trying to get to critical section', by having $(nt_i, nt_i, \circ)$ and $(t_ic, nt_i, \bullet)$, for $i = 1, 2$, controlling that. Initially we have both $(cn, t_1c, \bullet)$ and $(cn, t_2c, \bullet)$ on. One way to avoid the loss of liveness is to have that when a process moves to t_i then $(cn, t_jc, \bullet)$ (for $j = 3 - i$) becomes off, that is we have the switch $(nt_i, cn, t_jc, \bullet\circ)$. Clearly when that process moves out of t_i the restriction is not needed anymore, and so we have $(t_ic, cn, t_jc, \bullet\bullet)$. This constitutes a complete solution for $N = 2$ but for $N > 2$ it loses liveness and non-blocking. If we want to accommodate more processes we can simply add more trying states, as many as the number of processes being considered and have the same switch structure between every pair t_i, t_j for $i \neq j$. Meaning that a solution for arbitrary N is given by $S_N = (W_N, R_N)$ with $W_N = \{n, c, t_1, \ldots, t_N\}$ and

$$\begin{aligned} R_N \quad = \quad & \{(n, t_i), (t_i, c), (c, n), \\ & (nt_i, nt_i, \circ), (t_ic, nt_i, \bullet), (cn, t_ic, \bullet), (cn, t_jc, \bullet), (t_ic, t_jc, \circ), \\ & (nt_i, cn, t_jc, \bullet\circ), (t_ic, cn, t_jc, \bullet\bullet) : 1 \leq i, j \leq N, i \neq j\} \end{aligned}$$

Clearly for $N > 2$, S_N is not as easily visualised, but this is also a cost to pay also when drawing the transition system associated to the protocol for the mutual exclusion with various processes. We believe that it is still impressive the fact that the switch structure is easily definable for all N, being clear from the $N = 2$ case analysis why it works in the general case.

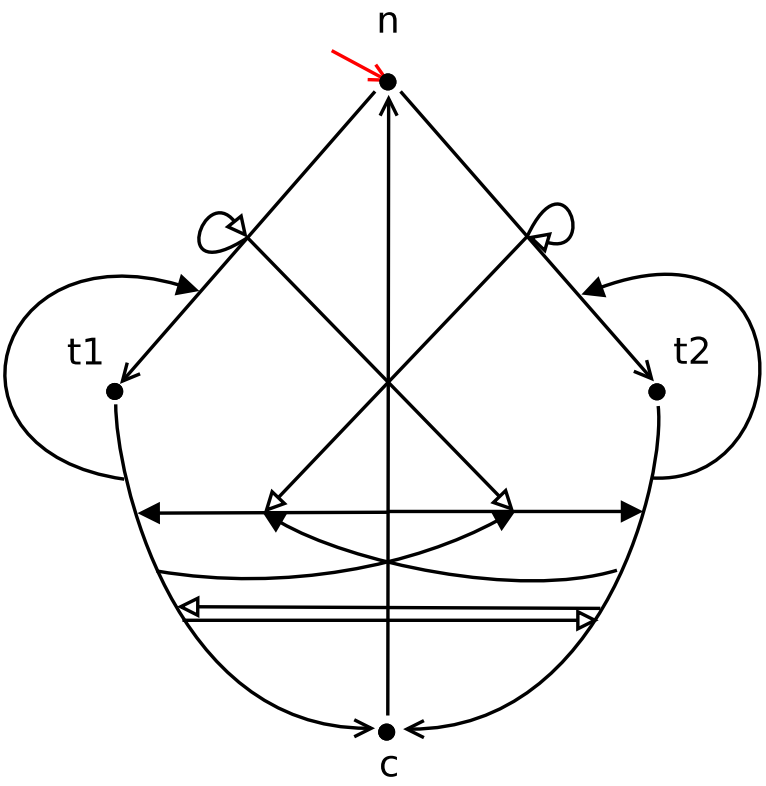

Figure 12: N arbitrary: safety, liveness, no strict sequencing and non-blocking for $N = 2$.

2.3 Comments and remarks

Level of Reactivity Being true that all reactive graphs can be represented by switch graphs, the level of reactivity of the representation is of obvious practical importance.

Given a reactive graph, it is easy to see that the minimal reactive level for a switch graph representing it, depends on the choice of dynamics we discussed above. Consider $R = (\{a,b\},\Delta)$, where $\Delta = \{\alpha : \lambda = (a,a)^n$ or $\alpha = (a,a)^{2n}(a,b), n < \omega\}$. Using the definition where the connecting switches win we cannot find a finite reactive level switch graph that represents it. Whereas using the alternating one it can be represented by a switch graph of reactivity level 1:

$$(\{a,b\},\{(a,a),(a,b),((a,a),(a,b),\circ),((a,a),(a,b),\bullet)\}).$$

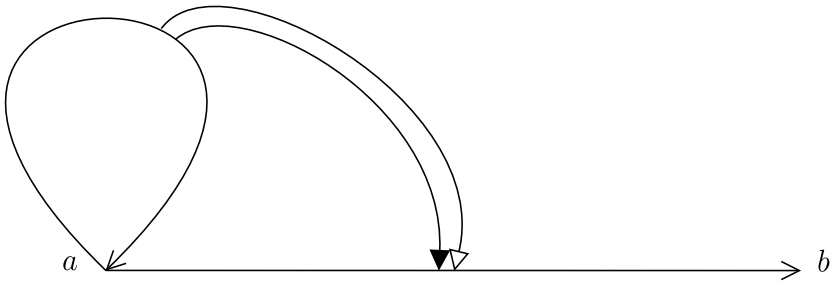

Figure 13: Alternating dynamics example with the following behaviour: $\Delta = \{\alpha : \lambda = (a,a)^n$ or $\alpha = (a,a)^{2n}(a,b), n < \omega\}$.

It is also easy to cook a reactive graph that cannot be represented by a finite level graph for any of the options we mentioned above, nor with any switches with 'recursive behaviour'. Let f be a non recursive binary sequence and

$R = (\{a,b\},\Delta)$, where $\Delta = \{\alpha : \alpha = (a,a)^n$ or $\alpha = (a,a)^k(a,b), n < \omega, f(k) = 1\}$.

Question 2.14. *For each of the considered options which is the set of reactive graphs generated by switch graphs of reactivity level $n \in \omega$?*

Further extensions We have seen how arrows can represent more than the basic transitions, they can represent the transitions between states of the accessibility relation itself. These new arrows are a natural extension of the concept of graph corresponding to a kind of meta-transitions in the above sense. They provide an explicit way of expressing the meta-level graph's notion of reactivity, in a way that allows an immediate (and complete) reading of the effects of crossing an edge.

In Section 2.2 we considered multiple entities going through the graph, the effects of each action were independent of whom was doing it and no synchronous movements were allowed. Other types of dependences can be considered:

- Dependence on the identity of the individual entities could be modelled by having different relations for each entity, or groups of entities.

- Synchronous movements could also be represented by special relations. Of course switches can connect arrows regardless of the relation they represent.

Thus leading to the question:

Question 2.15. *Is the switches formalism enough to represent these more general behaviours?*

Model Checking Above we gave an example of how a mutual exclusion protocol may be coded by a switch graph. Specifically, we model at the 'Kripke level', how the action of one agent (i.e. the transition between worlds/states) precludes the execution of an action (i.e. transition) of another agent. In this case switches are associated to changes on the auxiliary variables keeping the critical area safe. It is clear that the fact that we directly express that when one process access to the critical area, all the others will not be able to access to it (by having the appropriate switch in place, and no other switch acting on it), may give enough evidence to guaranty safety, without the need to check the whole state space.
It may be that depending on the property in question, the most effective way of verifying a certain system satisfies involves considering different types of models associated to that system, and checking a stronger property that entails it. Hopefully switch graphs are among the useful structures to do it. In the future we hope to find more meaningful examples that allow us to understand the general case.

3 Reactive hybrid switch logics

In the previous section we have shown that switch graphs are suitable structures to embody and represent the reactive paradigm. In this section we introduce a logic that allows us to reason about switches and their effects and prove completeness. A basic feature of such a logic would be to be able to express that if a certain switch is on, then after such move the state of certain switch will be such.

In the introduction we gave some motivation for the used language. On one hand, given a switch graph (W,R), for each switch type $\sigma \in \{\circ,\bullet\}^*$, the set of σ-switches forms $|\sigma|+2$-ary relation over W and there is the $|\sigma|+1$-ary modal correspondent, $\diamondsuit_\sigma$. On the other hand, similarly to $\diamondsuit_R$ in [19], there is the operator corresponding to its behaviour, $\diamondsuit$. The goal is to relate both, expressing how the switches determine the behaviour.

The logics we considered in [19] were suitable to talk about local effects of reactivity, but here we are dealing with its global effects and at the same time we feel the need to to explicitly refer to specific states. Ordinary modal logic's lack of mechanisms for dealing with states explicitly is a recognised weakness. In fact the idea of adding variables that are used to name worlds dates back to the pioneering work of Prior [30, 29] and especially of Bull [11](see [24]). Recently the study of this idea gained popularity and its development became an autonomous subfield of modal logic, called *hybrid logic*[2].

[2]For more details see [2, 1, 6].

Hybrid languages are a very simple extension of modal ones. A special set of propositional variables, called nominals, is added and it is used to name worlds. Nominals are true at exactly one world in any model. Many operators related to the nominals were studied. A particularly simple one is @ that allows one to jump to particular worlds, in the sense that, for each nominal i, $@_i\varphi$ is true if φ is true in the world named i, which suits our need to think globally.

We consider models constructed from switch graphs in such a way that for each admissible sequence of edges we get an usual hybrid model based on the Kripke frame given by the switches' state at that point. Of course each nominal must be true in the same world in all these models but all the other variables may change.

This language allows us to express some strong reactive assertions. Consider the formula

$$@_i \Diamond (j \to @_i \neg \Diamond j).$$

This says that if we cross the edge from the world named i to the world named j, that connection will be turned off. Another way to say this is to say that there is a disconnecting switch from that edge to itself:

$$@_i \Diamond_\circ (j, i, j).$$

We obtain completeness for the introduced logic adapting the usual Henkin style proof of completeness for hybrid logics, see [7, 9]. This further justifies the choice of language by showing that it is appropriate to capture the switches' dynamics.

3.1 Switch Models

Definition 3.1. We consider the *reactive switch similarity type*

$$s = (\{\Diamond\} \cup \{\Diamond_\sigma : \sigma \in \{\circ, \bullet\}^*\}, \rho),$$

where $\rho(\Diamond) = 1$ and $\rho(\Diamond_\sigma) = 2|\sigma| + 1$. Thus we define the hybrid modal language $\mathcal{H}_s(@)$ is defined by

$$\varphi ::= i \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \Diamond\varphi \mid \Diamond_\sigma(\varphi_1, \ldots, \varphi_{2|\sigma|+1}) \mid @_i\varphi,$$

where $p \in \Pi$, $i \in NOM$ and $\sigma \in \{\circ, \bullet\}^*$. The other connectives: $\top$, $\bot$, $\vee$, $\to$, $\leftrightarrow$, $\boxtimes$ and $\square_\sigma$ are introduced by the usual abbreviations.

Given $\lambda \in (NOM \times NOM)^*$ we define the abbreviation $\Diamond^\lambda\varphi$ by recursion:

- $\Diamond^\epsilon\varphi = \varphi$,
- $\Diamond^{(i,j)\lambda}\varphi = @_i \Diamond (j \wedge \Diamond^\lambda\varphi)$ (clearly $\Diamond^{(i,j)\lambda}\varphi = \Diamond^{(i,j)} \Diamond^\lambda \varphi$),

and $\boxtimes^\lambda\varphi = \neg \Diamond^\lambda \neg\varphi$.

Definition 3.2. Given a switch graph $S = (W, R)$ and using the dynamics defined in 2.6 we generate, for each $\lambda \in \Delta_S$, a reactive graph: $S_\lambda = (W, R_\lambda)$.

A switch frame is the Kripke frame given by the disjoint union of all these switch graphs, where σ switches in R_λ give origin to local $(2|\sigma| + 2)$-ary relations and with a global accessibility relation connecting w in S_λ and w' in $S_{\lambda(w,w')}$, see Figure 14.

Formally, the **switch frame** over S is $\mathfrak{F}_S = (W \times \Delta_S, \check{R}_S)$ where

- $\check{R}_S = \{\check{R}^\sigma_\lambda\}_{\sigma\in\{\circ,\bullet\}^*,\lambda\in\Delta_S}$ and
- $((w_1,\lambda),\ldots,(w_{2n+2},\lambda)) \in \check{R}^\sigma_\lambda$ iff $(w_1w_2,\ldots,w_{2n+1}w_{2n+2},\sigma) \in R_\lambda$.

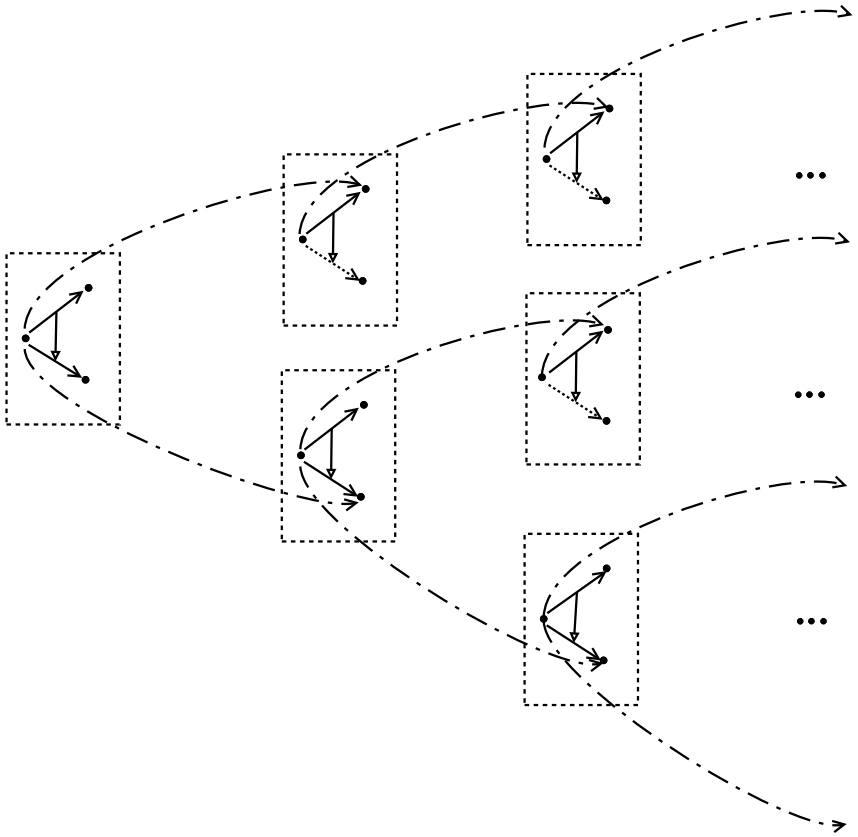

Figure 14: A representation of a switch frame. The various components are limited by a dashed line, the switches of that component represented by the arrows as before and the (reactive) transitions by the line formed by lines and dots.

A *switch model* over $\mathfrak{F}_S$ is a pair $\mathfrak{M} = (\mathfrak{F}, \nu)$, where ν is a function

$$\nu : \Pi \cup NOM \to 2^{\Delta_S \times W}$$

such that for $s \in NOM$ we have $V(s) = \{w\} \times \Delta_S$ for some $w \in W$.

Given a switch model $\mathfrak{M} = (W \times \Delta_S, R_S, \nu)$, for every $(w,\lambda) \in W \times \Delta_S$ and every L-formula φ, we define the notion φ *is true at* (w,λ) *in* $\mathfrak{M}$ $(\mathfrak{M},(w,\lambda) \vDash \varphi)$ inductively as follows:

- $\mathfrak{M},(w,\lambda) \vDash p$ iff $(w,\lambda) \in \nu(p)$ for variables p,

- $\mathfrak{M}, (w, \lambda) \vDash s$ iff $(w, \lambda) \in \nu(s)$ for nominals s,
- $\mathfrak{M}, (w, \lambda) \vDash \neg\varphi$ iff $\mathfrak{M}, (w, \lambda) \nvDash \varphi$,
- $\mathfrak{M}, (w, \lambda) \vDash \varphi_1 \wedge \varphi_2$ iff $\mathfrak{M}, (w, \lambda) \vDash \varphi_1$ and $\mathfrak{M}, (w, \lambda) \vDash \varphi_2$,
- $\mathfrak{M}, (w, \lambda) \vDash \Diamond\varphi$ iff there is $w' \in W$ such that $\lambda(w, w') \in \Delta$ and $\mathfrak{M}, (w', \lambda(w, w')) \vDash \varphi$,
  Notice that $\lambda(w, w')$ iff $(w, w') = (w, w', \epsilon) \in R_\lambda$ iff $(w, w') \in \check{R}^{\epsilon}_{\lambda}$,
- $\mathfrak{M}, (w, \lambda) \vDash @_i\varphi$ iff $\mathfrak{M}, (w', \lambda) \vDash \varphi$ for $\nu(i) = \Delta_S \times \{w'\}$,
- $\mathfrak{M}, (w, \lambda) \vDash \Diamond_\sigma(\varphi_1, \ldots, \varphi_{2n+1})$ iff there are $v_1 \ldots v_{2n+1} \in W$ such that $(w, v_1, \ldots, v_{2n}, v_{2n+1}) \in \check{R}^{\sigma}_{\lambda}$ and $\mathfrak{M}, (\lambda, v_i) \vDash \varphi_i$.

We say that φ *is true in* $\mathfrak{M}$ iff $\mathfrak{M}, (w, \lambda) \vDash \varphi$ for every $(w, \lambda) \in \Delta_S \times W$. We say that φ *is valid in* a switch frame if it is true in every switch model over it.

3.2 Axiomatising

In this section we prove that the axiomatisation presented in Figure 15, generates all the $\mathcal{H}_s(@)$-formulas that are valid in all switch frames. Furthermore, we obtain the usual hybrid automatic completeness for pure axioms. The following axiomatisation is the natural adaptation of the ones for the standard hybrid logics given in [8, 9].

Remark 3.3. Notice that if we considered a different dynamics for the switches, *Dyn* would have to be different. For example the alternatives in 2.7 correspond to:

- the disconnecting switch would prevail
  $\Diamond^{(i,j)} @_{i_1} \Diamond_\sigma (i_2, \ldots, i_{2n+2}) \leftrightarrow$
  $\neg @_i \Diamond_{\sigma\circ} (j, i_1 \ldots, i_{2n+2}) \wedge (@_{i_1} \Diamond_\sigma (i_2, \ldots, i_{2n+2}) \vee @_i \Diamond_{\sigma\bullet} (j, i_1 \ldots, i_{2n+2}))$,
- always changing state
  $\Diamond^{(i,j)} @_{i_1} \Diamond_\sigma (i_2, \ldots, i_{2n+2}) \leftrightarrow$
  $(@_{i_1}\Diamond_\sigma(i_2, \ldots, i_{2n+2}) \wedge \neg @_i \Diamond_{\sigma\circ}(j, i_1 \ldots, i_{2n+2}) \vee$
  $\neg @_{i_1} \Diamond_\sigma (i_2, \ldots, i_{2n+2}) \wedge @_i \Diamond_{\sigma\bullet} (j, i_1 \ldots, i_{2n+2}))$.

Here is the theorem we shall prove:

Theorem 3.4. *Let Λ be a set of pure $\mathcal{H}_s(@)$ formulas. A set of $\mathcal{H}_s(@)$ formulas Σ is $L_s + \Lambda$-consistent iff Σ is satisfiable in a model satisfying the frame properties defined by Γ. Where $L_s + \Lambda$ is the above axiomatisation extended with the axioms of Λ.*

Lemma 3.5. *The following are derivable*

- $K^{-1}_{@}$: $\vdash (@_i\varphi \to @_i\psi) \to @_i(\varphi \to \psi)$,
- *Nom:* $\vdash @_i j \to (@_i\varphi \to @_j\varphi)$,
- *Sym:* $\vdash @_i j \to @_j i$,

L_s	
Axioms:	
CT	All classical tautologies
$K^l_{\Box_\sigma}$	$\Box_\sigma(\varphi_1,\ldots,\varphi\to\psi,\ldots,\varphi_{2n+1})\to$ $(\Box_\sigma(\varphi_1,\ldots,\varphi,\ldots,\varphi_{2n+1})\to\Box_\sigma(\varphi_1,\ldots,\psi,\ldots,\varphi_{2n+1}))$ where l is the component where $\varphi\to\psi$ is.
$K_@$	$@_i(\varphi\to\psi)\to(@_i\varphi\to@_i\psi)$
$Selfdual_@$	$@_i\varphi\leftrightarrow\neg@_i\neg\varphi$
$Ref_@$	$@_i i$
$Agree$	$@_i@_j\varphi\leftrightarrow@_j\varphi$
$Intro$	$i\to(\varphi\leftrightarrow@_i\varphi)$
$Sync$	$\Diamond i\leftrightarrow\Diamond_\epsilon i$
$Det_\Diamond$	$\Diamond(i\wedge\varphi)\to\boxslash(i\to\varphi)$
$K_\boxslash$	$\boxslash(\varphi\to\psi)\to(\boxslash\varphi\to\boxslash\psi)$
Dyn	$\Diamond^{(i,j)}@_{i_1}\Diamond_\sigma(i_2,\ldots,i_{2n+2})\leftrightarrow$ $(@_{i_1}\Diamond_\sigma(i_2,\ldots,i_{2n+2})\wedge\neg@_i\Diamond_{\sigma\circ}(j,i_1\ldots,i_{2n+2})\vee@_i\Diamond_{\sigma\bullet}(j,i_1\ldots,i_{2n+2}))$
Rules:	
MP	If $\vdash\varphi$ and $\vdash\varphi\to\psi$ then $\vdash\psi$
$Subst$	If $\vdash\varphi$ then $\vdash\varphi^\theta$
$Gen_{\Box_\sigma}$	If $\vdash\varphi$ then $\vdash\Box_\sigma(\bot,\ldots,\bot,\varphi,\bot,\ldots,\bot)$
$Gen_@$	If $\vdash\varphi$ then $\vdash@_i\varphi$
$Gen_\boxslash$	If $\vdash\varphi$ then $\vdash\boxslash\varphi$
$Name$	If $\vdash@_i\varphi$ and i does not occur in φ then $\vdash\varphi$
$Paste_{\Diamond^\lambda\Diamond_\sigma}$	If $\vdash\Diamond^\lambda@_i\Diamond_\sigma(j_1,\ldots,j_{2n+1})\wedge\Diamond^\lambda(\bigwedge_{1\geq l\geq 2n+1}@_{j_l}\varphi_l)\to\psi$ and $j_m\neq i$ does not occur in φ_l or ψ then $\vdash\Diamond^\lambda@_i\Diamond_\sigma(\varphi_1,\ldots,\varphi_{2n+1})\to\psi$
$Paste'_{\Diamond^\lambda\Diamond}$	If $\vdash\Diamond^\lambda@_i\Diamond j\wedge\Diamond^\lambda\Diamond@_j\varphi\to\psi$ and $j\neq i$ does not occur in φ or ψ then $\vdash\Diamond^\lambda@_i\Diamond\varphi\to\psi$

Figure 15: The L_s axiomatisation.

- *Name′: If* $\vdash i\to\varphi$ *then* $\vdash\varphi$ *where* i *does not occur in* φ,
- $Back_{\Diamond_\sigma}$: $\Diamond_\sigma(\varphi_1,\ldots,@_i\varphi_k,\ldots,\varphi_{2n+1})\to@_i\varphi_k$,
- $Bridge_{\Diamond_\sigma}$: $@_i\Diamond_\sigma(j_1,\ldots,j_{2n+1})\wedge(\bigwedge_{1\geq l\geq 2n+1}@_{j_l}\varphi_l)\to@_i\Diamond_\sigma(\varphi_1,\ldots,\varphi_{2n+1})$.

Proof. The first 4 are proved in [9].

- $Back_\Diamond$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{@_i@_j\varphi\leftrightarrow@_u@_j\varphi}\,Agree,Agree}{@_u\Diamond_\sigma(\varphi_1,\ldots,j,\ldots,\varphi_{2n+1})\wedge@_i@_j\varphi\to@_u@_j\varphi}\,Paste_{\Diamond_\sigma}}{@_u\Diamond_\sigma(\varphi_1,\ldots,@_j\varphi,\ldots,\varphi_{2n+1})\to@_u@_j\varphi}\,K_@^{-1},Name}{\Diamond_\sigma(\varphi_1,\ldots,@_j\varphi,\ldots,\varphi_{2n+1})\to@_j\varphi}$$

- $Bridge_{\Diamond^\sigma}$ by induction on σ, $\sigma=\epsilon$ is trivial and

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Box_\epsilon(\neg\varphi \to \neg j) \to (\Box_\epsilon\neg\varphi \to \Box_\epsilon\neg j)}\,K_{\Box_\epsilon}}{\Box_\epsilon\neg\varphi \wedge \Diamond_\epsilon j \to \Diamond_\epsilon(j \wedge \neg\varphi)}\,CT \quad \dfrac{}{j \wedge \neg\varphi \to @_j\neg\varphi}\,Int_@, CT}{\Box_\epsilon\neg\varphi \wedge \Diamond_\epsilon j \to \Diamond_\epsilon @_j\neg\varphi}\,CT \quad \dfrac{}{\Diamond_\epsilon @_j\neg\varphi \to @_j\neg\varphi}\,Back}{@_i(\Box_\epsilon\neg\varphi \wedge \Diamond_\epsilon j) \to @_i@_j\neg\varphi}\,Gen_@, K_@}{@_i\,\Box_\epsilon\,\neg\varphi \wedge @_i\,\Diamond_\epsilon\,j \to @_j\neg\varphi}\,K_@, CT, Agree, Self dual_@}{@_i\,\Diamond_\epsilon\,j \wedge @_j\varphi \to @_i\,\Diamond_\epsilon\,\varphi}\,CT$$

The proof for arbitrary σ's consists in the repetition of this derivation applied to each of the $2|\sigma| - 1$ coordinates, where in each time an argument eats a conjunct.

□

Lemma 3.6. *The following are derivable:*

- $\boxtimes^{(i,j)\lambda}\varphi \leftrightarrow @_i \boxtimes (j \to \boxtimes^\lambda\varphi)$,
- $Gen_{\boxtimes^\lambda}$*: If* $\vdash \varphi$ *then* $\vdash \boxtimes^\lambda\varphi$,
- $K_{\boxtimes^\lambda}$*:* $\boxtimes^\lambda(\varphi \to \psi) \to (\boxtimes^\lambda\varphi \to \boxtimes^\lambda\psi)$,
- $Det_{\Diamond^\lambda}$*:* $\Diamond^\lambda\varphi \leftrightarrow \boxtimes^\lambda\varphi \wedge \Diamond^\lambda\top$,
- $Gen'_{\Diamond^\lambda}$*:* $\vdash \varphi \to \psi$ *then* $\vdash \Diamond^\lambda\varphi \to \Diamond^\lambda\varphi$.

Furthermore from $Gen_{\boxtimes^\lambda}$ *and* $K_{\boxtimes^\lambda}$ *we easily get* $\Diamond^\lambda(\varphi \wedge \psi) \leftrightarrow \Diamond^\lambda\varphi \wedge \Diamond^\lambda\psi$ *and* $\Diamond^\lambda(\varphi \vee \psi) \leftrightarrow \Diamond^\lambda\varphi \vee \Diamond^\lambda\psi$.

Proof.

- $\boxtimes^{(i,j)\lambda}\varphi = \neg@_i \Diamond (j \wedge \Diamond^\lambda\neg\varphi)$ that is equivalent to $@_i \boxtimes (j \to \neg \Diamond^\lambda \neg\varphi) = @_i \boxtimes (j \to \boxtimes^\lambda\varphi)$.
- $Gen_{\boxtimes^\lambda}$ by induction on λ, ϵ is trivial and

$$\dfrac{\dfrac{\dfrac{\dfrac{\varphi}{\boxtimes^\lambda\varphi}\,IH}{j \to \boxtimes^\lambda\varphi}\,CT}{\boxtimes(j \to \boxtimes^\lambda\varphi)}\,Gen_\boxtimes}{@_i \boxtimes (j \to \boxtimes^\lambda\varphi)}\,Gen_@$$

- $K_{\boxtimes^\lambda}$ by induction on λ, $\lambda = \epsilon$ is trivial and

$$\dfrac{\dfrac{\dfrac{\dfrac{\varphi}{\boxtimes^{(i,j)\lambda}(\varphi \to \psi)}\,IH}{@_i \boxtimes (j \to (\boxtimes^\lambda\varphi \to \boxtimes^\lambda\psi))}\,CT}{@_i \boxtimes (j \to \boxtimes^\lambda\varphi) \to @_i \boxtimes (j \to \boxtimes^\lambda\psi)}\,K_\boxtimes, K_@}{\boxtimes^{(i,j)\lambda}\varphi \to \boxtimes^{(i,j)\lambda}\psi}\,def$$

- $Det_{\Diamond^{(i,j)\lambda}}$ by induction on λ, $\lambda = \epsilon$ is trivial and

$$\dfrac{\dfrac{}{\boxdot^{(i,j)\lambda}\varphi \wedge \Diamond^{(i,j)\lambda}\top \rightarrow \Diamond^{(i,j)\lambda}\varphi}\ K_{\boxdot}^{(i,j)\lambda}, CT \qquad \dfrac{\dfrac{}{\Diamond^{(i,j)\lambda}\varphi \rightarrow @_i \Diamond (j \wedge [\boxdot^{\lambda}\varphi \wedge \Diamond^{\lambda}\top])}\ Def, IH}{\Diamond^{(i,j)\lambda}\varphi \rightarrow @_i \Diamond (j \wedge \boxdot^{\lambda}\varphi)}\ IH, CT}{\Diamond^{(i,j)\lambda}\varphi \rightarrow \boxdot^{(i,j)\lambda}\varphi \wedge \Diamond^{(i,j)\lambda}\top}\ CT$$
$$\Diamond^{(i,j)\lambda}\varphi \leftrightarrow \boxdot^{(i,j)\lambda}\varphi \wedge \Diamond^{(i,j)\lambda}\top \qquad CT$$

- $Gen'_{\Diamond^{\lambda}}$ by induction on λ, $\lambda = \epsilon$ is trivial and

$$\dfrac{\dfrac{\dfrac{\dfrac{\varphi \rightarrow \psi}{\boxdot^{\lambda}(\varphi \rightarrow \psi)}\ Gen_{\boxdot^{\lambda}}}{\Diamond^{\lambda}\top \wedge \boxdot^{\lambda}(\varphi \rightarrow \psi)}\ CT}{\boxdot^{\lambda}\varphi \wedge \Diamond^{\lambda}\top \rightarrow \boxdot^{\lambda}\psi \wedge \Diamond^{\lambda}\top}\ K_{\boxdot^{\lambda}}, CT}{\Diamond^{\lambda}\varphi \rightarrow \Diamond^{\lambda}\psi}\ Det_{\Diamond^{\lambda}}$$

□

Definition 3.7. *Let Σ be a set of $\mathcal{H}_s(@)$-formulas:*

- *Σ is named if one of its elements is a nominal.*
- *Σ is $\Diamond\!\!\!\!\!\Diamond_\sigma$-saturated if for all σ' and $\Diamond^{\lambda}@_i \Diamond\!\!\!\!\!\Diamond_{\sigma'} (\varphi_1, \ldots, \varphi_{2n+1}) \in \Sigma$ there are nominals $j_1, \ldots, j_{2n+1}$ such that $\Diamond^{\lambda}@_i \Diamond\!\!\!\!\!\Diamond_{\sigma'} (j_1, \ldots, j_{2n+1}) \in \Sigma$ and $\Diamond^{\lambda}@_{j_k}\varphi_k \in \Sigma$, $k = 1, \ldots, 2n+1$.*
- *Σ is $\Diamond$-saturated if for all $\Diamond^{\lambda}@_i \Diamond \varphi \in \Sigma$ there is a nominal j such that $\Diamond^{\lambda}@_i \Diamond (j \wedge \varphi) = \Diamond^{\lambda(i,j)}\varphi \in \Sigma$.*

Lemma 3.8. *(Lindenbaum Lemma). Every $L_s + \Gamma$-consistent set of formulas can be extended to a named, $\Diamond$-saturated and $\Diamond\!\!\!\!\!\Diamond_\sigma$-saturated MCS, by adding countably many new nominals to the language.*

Proof. Let $(i_n)_{n<\omega}$ be an enumeration of the new nominals and $(\varphi_n)_{n<\omega}$ an enumeration of the formulas in the extended language.

We define $\Sigma^0 = \Sigma \cup i_0$, *Name'* guarantees that it is consistent.

If $\Sigma^n \cup \{\varphi_n\}$ is inconsistent then $\Sigma^{n+1} = \Sigma^n$. Otherwise:

1. $\Sigma^{n+1} = \Sigma^n \cup \{\varphi_n\}$ if φ_n is not of the form $\Diamond^{\lambda}@_i \Diamond\!\!\!\!\!\Diamond_\sigma (\psi_1, \ldots, \psi_{2k+1})$ or $\Diamond^{\lambda}@_i \Diamond \psi$,
2. $\Sigma^{n+1} = \Sigma^n \cup \{\varphi_n\} \cup \{\Diamond^{\lambda}@_i \Diamond\!\!\!\!\!\Diamond_\sigma (i_m, \ldots, i_{m+2n+1})\} \cup \{\Diamond^{\lambda}@_{i_{m+l}}\psi_l : 1 \leq l \leq 2k+1\}$ if it is of the form $\Diamond^{\lambda}@_i \Diamond\!\!\!\!\!\Diamond_\sigma (\psi_1, \ldots, \psi_{2k+1})$,
3. $\Sigma^{n+1} = \Sigma^n \cup \{\varphi_n\} \cup \{\Diamond^{\lambda}@_i \Diamond (i_m \wedge \psi)\}$ if it is of the form $\Diamond^{\lambda}@_i \Diamond \psi$.

Where i_m is the first new nominal that does not occur in Σ^n or φ_n.
Let $\Sigma^\omega = \bigcup_{n<\omega} \Sigma^n$. Then $\Sigma \subseteq \Sigma^\omega$ and Σ^ω is named, $\Diamond\!\!\!\!\!\Diamond_\sigma$-saturated, $\Diamond$-saturated, maximal and consistence. The only non-trivial step is in 2, and here consistency is guaranteed by $Paste_{\Diamond^{\lambda}\Diamond\!\!\!\!\!\Diamond_\sigma}$ and $Paste'_{\Diamond^{\lambda}\Diamond}$. □

Definition 3.9. *(Henkin model from Γ). Let Γ be a maximal consistent set of $\mathcal{H}_s(@)$ formulas. For all nominals i, let $|i| = \{j : @_i j \in \Gamma\}$. Then the $\mathfrak{M}_\Gamma = (W, \Delta, T, \nu)$ is given by:*

$$
\begin{aligned}
&W = \{|i| : i \text{ is a nominal}\},\\
&\Delta = \{\langle\lambda\rangle : \Diamond^\lambda \top \in \Gamma\},\\
&\check{R} = \{T^\sigma_{\langle\lambda\rangle}\}_{\sigma,\langle\lambda\rangle},\\
&\check{R}^\sigma_{\langle\lambda\rangle}(|i_1|,\ldots,|i_{2n+2}|) \text{ iff } \Diamond^\lambda @_{i_1} \Diamond_\sigma (i_2,\ldots,i_{2n+2}) \in \Gamma,\\
&\nu(p) = \{(|i|,\langle\lambda\rangle) : \Diamond^\lambda @_i p \in \Gamma\},\\
&\nu(i) = \{(|i|,\langle\lambda\rangle) : \langle\lambda\rangle \in \Delta\},
\end{aligned}
$$

where $\langle\epsilon\rangle = \epsilon$ and $\langle\lambda(i,j)\rangle = \langle\lambda\rangle(|i|,|j|)$. That $\mathfrak{M}_\Gamma$ is well-defined follows from Ref, Sym, Nom and $Gen'_{\Diamond_\lambda}$.

Lemma 3.10. $\Diamond^{\lambda(i,j)}\varphi = \Diamond^\lambda @_i \Diamond (j \wedge \varphi)$.

Proof. By induction on λ:

For $\lambda = \epsilon$ the two formulas coincide trivially and for $\lambda = (s,t)\gamma$

$$\Diamond^{(s,t)\gamma(i,j)}\varphi = \Diamond^{(s,t)}\Diamond^{\gamma(i,j)}\varphi =^{IH} \Diamond^{(s,t)}\Diamond^\gamma @_i \Diamond (j \wedge \varphi) = \Diamond^{(s,t)\gamma} @_i \Diamond (j \wedge \varphi).$$

□

Lemma 3.11. *$\mathfrak{M}_\Gamma$ is a switch model where $(W \times \Delta, \check{R})$ is the switch frame generated by $S = (W, R)$ such that*

$$R = \{(|i_1||i_2|,\ldots,|i_{2n+1}||i_{2n+2}|,\sigma) : (|i_1|,|i_2|,\ldots,|i_{2n+1}|,|i_{2n+2}|) \in \check{R}^\sigma_\epsilon\}.$$

Proof. Let Δ_S and $R_S = \{R^\sigma_\lambda\}$ as defined in 2.6. The proof that $\Delta = \Delta_S$ and $\check{R}^\sigma_\lambda = R^\sigma_\lambda$ is done by induction on the length of $\langle\lambda\rangle$.

The case of $\langle\lambda\rangle = \epsilon$ is trivial since it is in both Δ ($\top \in \Gamma$ by CT) and Δ_S, furthermore $\check{R}^\sigma_\epsilon = R^\sigma_\epsilon$ by definition.

The induction step ($\langle\lambda\rangle \to \langle\lambda(i,j)\rangle$):

$\langle\lambda(|i|,|j|)\rangle \in \Delta_S$ iff

$\langle\lambda\rangle \in \Delta_S$ and $(|i|,|j|) \in R_\lambda$ iff $\langle\lambda\rangle \in \Delta_S$ and $R^\epsilon_{\langle\lambda\rangle}(|i|,|j|)$ iff (IH)

$\Diamond^\lambda\top \in \Gamma$ and $\check{R}^\epsilon_{\langle\lambda\rangle}(|i|,|j|)$ iff $\Diamond^\lambda @_i \Diamond (j \wedge \top) = \Diamond^{\lambda(i,j)}\top \in \Gamma$ iff

$\langle\lambda\rangle(|i|,|j|) \in \Delta$.

And,

$(|i_1||i_2|,\ldots,|i_{2n+1}||i_{2n+2}|,\sigma) \in R_{\langle\lambda(i,j)\rangle}$ iff

$(|i_1||i_2|,\ldots,|i_{2n+1}||i_{2n+2}|,\sigma) \in R_{\langle\lambda\rangle}$ and $(|i||j|,|i_1||i_2|,\ldots,|i_{2n+1}||i_{2n+2}|,\sigma\circ) \notin R_{\langle\lambda\rangle}$

or

$(|i||j|,|i_1||i_2|,\ldots,|i_{2n+1}||i_{2n+2}|,\sigma\bullet) \in R_{\langle\lambda\rangle}$

iff (IH)

$\check{R}^\sigma_\lambda(|i_1|,|i_2|,\ldots,|i_{2n+1}|,|i_{2n+2}|,\sigma)$ and $\check{R}^{\sigma\circ}_\lambda(|i|,|j|,|i_1|,|i_2|,\ldots,|i_{2n+1}|,|i_{2n+2}|)$

or

$\check{R}^{\sigma\bullet}_{\lambda}(|i|,|j|,|i_1|,|i_2|,\ldots,|i_{2n+1}|,|i_{2n+2}|)$
iff
$\Diamond^{\lambda}@_{i_1}\Diamond_{\sigma}(i_2,\ldots,i_{2n+2})\in\Gamma$ and $\Diamond^{\lambda}@_i\Diamond_{\sigma\circ}(j,i_1\ldots,i_{2n+2})\notin\Gamma$
or
$\Diamond^{\lambda}@_i\Diamond_{\sigma\bullet}(j,i_1\ldots,i_{2n+2})\in\Gamma$
iff
$\Diamond^{\lambda}@_{i_1}\Diamond_{\sigma}(i_2,\ldots,i_{2n+2})\wedge\Diamond^{\lambda}\neg@_i\Diamond_{\sigma\circ}(j,i_1\ldots,i_{2n+2})\vee\Diamond^{\lambda}@_i\Diamond_{\sigma\bullet}(j,i_1\ldots,i_{2n+2})\in\Gamma$
iff (*Dyn*, $Gen'_{\Diamond^{\lambda}}$ and Lemma 3.6)
$\Diamond^{\lambda}\Diamond^{(i,j)}@_{i_1}\Diamond_{\sigma}(i_2,\ldots,i_{2n+2})\in\Gamma$
iff (Lemma 3.10)
$\Diamond^{\lambda(i,j)}@_{i_1}\Diamond_{\sigma}(i_2,\ldots,i_{2n+2})\in\Gamma$.

□

Lemma 3.12 (Truth lemma)**.** *For all $\Diamond$-saturated and $\Diamond_{\sigma}$-saturated $L_s+\Lambda$-MCS's Γ, nominals i and formulas φ,*

$$\mathfrak{M},(|i|,\langle\lambda\rangle)\vDash\varphi \textit{ iff } \Diamond^{\lambda}@_i\varphi\in\Gamma.$$

Proof. By induction on the length of φ.

- propositional symbols and nominals by definition;
- $\varphi=\neg\psi$
  $\mathfrak{M}_{\Gamma},(|i|,\langle\lambda\rangle)\vDash\neg\psi$ iff $\mathfrak{M}_{\Gamma},(|i|,\langle\lambda\rangle)\nvDash\psi$ and $\mathfrak{M}_{\Gamma},(|i|,\langle\lambda\rangle)\vDash\top$
  iff (IH)
  $\Diamond^{\lambda}@_i\psi\notin\Gamma$ and $\Diamond^{\lambda}@_i\top\in\Gamma$ iff $\neg\Diamond^{\lambda}@_i\psi\wedge\Diamond^{\lambda}\top\in\Gamma$
  iff ($Self dual_{@}$, $Gen'_{\Diamond^{\lambda}}$ and $Det_{\Diamond^{\lambda}}$)
  $\Diamond^{\lambda}@_i\neg\psi$;
- $\varphi=\psi_1\to\psi_2$ as in the previous case we apply $K_{@}$, $K^{-1}_{@}$, $Gen'_{\Diamond^{\lambda}}$ and $Det_{\Diamond^{\lambda}}$;
- $\varphi=@_j\psi$ apply *Agree* and $Gen'_{\Diamond^{\lambda}}$;
- $\varphi=\Diamond_{\sigma}\psi$;
  If $\mathfrak{M}_{\Gamma},(|i|,\langle\lambda\rangle)\vDash\Diamond_{\sigma}(\psi_1,\ldots,\psi_{2n+1})$ then there are $|j_1|,\ldots,|j_{2n+1}|$ such that
  $$\check{R}^{\sigma}_{\langle\lambda\rangle}(|i|,|j_1|,\ldots,|j_{2n+1}|)$$
  and $\mathfrak{M}_{\Gamma},(|j_l|,\langle\lambda\rangle)\vDash\psi_l$. By definition $\Diamond^{\lambda}@_i\Diamond_{\sigma}(j_1,\ldots,i_{2n+1})\in\Gamma$ and (IH) $@_{j_l}\psi_l\in\Gamma$ for $1\le l\le 2n+1$. Thus, using $Bridge_{\Diamond_{\sigma}}$, $Gen'_{\Diamond^{\lambda}}$ and Lemma 3.6, we have that
  $$\Diamond^{\lambda}@_i\Diamond_{\sigma}(\varphi_1,\ldots,\varphi_{2n+1})\in\Gamma.$$
  Conversely, suppose $\Diamond^{\lambda}@_i\Diamond_{\sigma}(\varphi_1,\ldots,\varphi_{2n+1})\in\Gamma$ then by $\Diamond^{\lambda}$-saturation there are $j_1,\ldots,j_{2n+1}$ such that $\Diamond^{\lambda}@_i\Diamond_{\sigma}(j_1,\ldots,j_{2n+1})\in\Gamma$ and $\Diamond^{\lambda}@_{j_l}\psi_l\in\Gamma$ for $1\le l\le 2n+1$. By IH $\mathfrak{M}_{\Gamma},(|j_l|,\langle\lambda\rangle)\vDash\psi_l$ and by definition $\check{R}^{\sigma}_{\langle\lambda\rangle}(|i|,|j_1|,\ldots,|j_{2n+1}|)$. Hence,
  $$\mathfrak{M}_{\Gamma},(|i|,\langle\lambda\rangle)\vDash\Diamond_{\sigma}(\psi_1,\ldots,\psi_{2n+1}).$$

- $\varphi = \Diamond\psi$

  $\mathfrak{M}_\Gamma, (|i|, \langle\lambda\rangle) \vDash \Diamond\varphi$ iff

  exists $|j|$ such that $\langle\lambda\rangle\,(|i|, |j|) \in \Delta$ such that $\mathfrak{M}_\Gamma, (|j|, \langle\lambda\rangle\,(|i|, |j|)) \vDash \Diamond\varphi$ iff (I.H.)

  exists $|j|$ such that $\Diamond^{\lambda(i,j)}@_j\varphi = \Diamond^\lambda@_i \Diamond\,(j \wedge @_j\varphi) \in \Gamma$ iff (using $\Diamond$-saturation and CT)

  $\Diamond^\lambda@_i \Diamond\varphi \in \Gamma$.

□

Lemma 3.13 (Frame lemma)**.** *For all $\Diamond$-saturated and $\Diamond\!\!\downarrow_\sigma$-saturated L_s + Λ-MCS's, $\mathfrak{M}_\Gamma$ satisfies the switch frame properties defined by Λ.*

Proof. It follows from the fact that $\Lambda \subseteq \Gamma$ contains $\boxslash^\lambda@_i\varphi$ where φ is an instance of an element of Λ from $Subst$, $Gen_@$ and $Gen_{\boxslash^\lambda}$. □

We can now prove Theorem 3.4.

Proof. (Of Theorem 3.4) Suppose Σ is L_s + Λ-consistent. By Lemma 3.8, Σ can be extended to a named, $\Diamond$-saturated and $\Diamond\!\!\downarrow_\sigma$-saturated L_s + Λ-MCS's Γ. Let $i \in \Sigma$. By Lemma 3.12 we have $\mathfrak{M}_\Gamma, (|i|, \langle\epsilon\rangle) \vDash \Sigma$. By Lemma 3.13, $\mathfrak{M}_\Gamma$ satisfies all required frame properties. □

3.3 Comments and remarks

Small switch frames The fact that the propositional valuation is allowed to change at each move implies that switch frames based on switch graphs with infinite behaviour have infinite components. Nevertheless, to have a complete view on the switch dynamics it would be enough to have one component for each reachable switch configuration. Let $A_S = \{R_\lambda : \lambda \in \Delta_S\}$ be the set of switches' configurations that are obtained in S's dynamics. One can then consider the Kripke frame $\mathfrak{F}'_S = (W \times A, \check{R}'_S)$ (the small switch frame over S) where

- $\check{R}'_S = \{\hat{R}^\sigma\}_{\sigma\in\{\circ,\bullet\}^*}$,
- $((w_1, R), \ldots, (w_{2n+2}, R)) \in \check{R}^\sigma_\lambda$ iff $(w_1w_2, \ldots, w_{2n+1}w_{2n+2}, \sigma) \in R$ and
- (w, R_λ) connects (reactively) with $(w', R_{\lambda ww'})$.

So everything is the same apart from the fact that we do not always get a new component from a move, the tree structure disappears. It is easy to see that $\mathfrak{F}'_S$ is finite if and only if S is finite. See Figure 16 for an example.

Let us consider a semantics of $\mathcal{H}_s(@)$ over the small switch frames trivially adapted from Definition 3.2 (only the interpretation of $\Diamond$ changes), and call them small switch models. If the set of propositional symbols is empty ($\Pi = \varnothing$) these two classes of frames trivially generate the same logic. Also, when, e.g. for modal checking purposes, everything in the model is finite, including the number of propositional variables

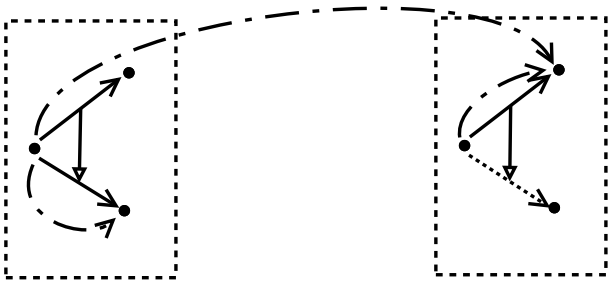

Figure 16: Small switch frame of $\{\{a,b,c\},\{(a,b),(a,c),(ab,cd,\circ)\}\}$.

involved, one would consider similarly a finite representation by having one component for each configuration of the switches and distribution of the relevant propositional variables.

Moreover, it is interesting to notice that even if the classes of small and the original switch frames are not immediately reducible to one another they yield the same logic.

It is trivial to define a model over $\mathfrak{F}_S$ from a model over $\mathfrak{F}'_S$ preserving modal truths, since we just have to copy the valuation of each configuration and paste it where it appears, so

$$\{\varphi \in \mathcal{H}_s(@) : \varphi \text{ is valid over switch frames}\} \subseteq \{\varphi \in \mathcal{H}_s(@) : \varphi \text{ is valid over small switch frames}\}.$$

The converse is not so immediate since if we cannot fit two different configurations in the same component of a small switch model. However, it is not hard to see that these two classes define the same logic. The idea of the proof is the following:

Given φ and a switch model $(W \times \Delta_S, R_S, \nu), (w,\epsilon) \vDash \varphi$ (we can assume $\lambda = \epsilon$ since the past clearly does not interfere with the evaluation of φ). We then consider a switch graph S' which is a copy of S plus some switches of a high enough level so that they do not interfere with the evaluation of φ and that during the evaluation of φ the new switches' configuration changes and so that we can accommodate the ν. Let k be the higher level of the switches referred in φ, it is clear we can add new points and switches such that at each crossing of an edge a different switch, of level higher than k, changes its state. For example one could add for each admissible sequence of edges covered in the interpretation of $\Diamond$, $\lambda = (w_1w_2)\dots(w_{2n-1}w_{2n})$, a point w_λ and the switch

$$(w_1w_2,\dots,w_{2n-1}w_{2n},[w_\lambda w_\lambda]^k,\bullet^{n-1+k})$$

where $[w_\lambda w_\lambda]^k$ stands for $k+1$-occurrences of $w_\lambda w_\lambda$ with a comma dividing each occurrence. Thus guaranteeing a different component for each sequence of edges and therefore being able to define the appropriate valuation to each of them.

Decidability and f.m.p. Given a satisfiable formula φ, that is, such that for some $(w,\lambda) \in W \times \Delta_S$ and valuation μ, we have:

$$(W \times \Delta_S, R_S, \mu), (w,\lambda) \vDash \varphi.$$

Clearly this is determined by just a part of the whole model. Has we have noticed above we can start by throwing out the past, that is, assuming that $\lambda = \epsilon$. It is also clear that only the fragment corresponding to the paths bounded by the $\Diamond$-modal nesting-depth of φ, $md_{\Diamond}(\varphi)$, matters. Furthermore, the level of the relevant switches is bounded by $n + md_{\Diamond}(\varphi)$, where n is the level of the highest switch in the formula. Thus, the satisfiability of φ is reduced to checking the satisfiability over certain Kripke models corresponding to that kind of fragment. Of course, if W is infinite these fragments are infinite. The most direct way to finitise such a fragment would be to adapt the filtration method by identifying only the points that satisfy the same relevant formulas in all of the fragment components.

We start by noticing that this method fails over general switch models. Consider the following switch graph

$$S = (\omega, \{(0, n), (0n, nn, \circ) : n < \omega\}),$$

see Figure 17, and $\Diamond \Diamond \top \in \mathcal{H}_s(@)$.

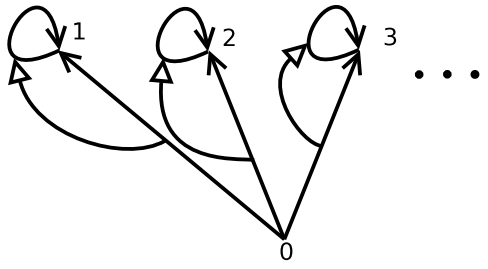

Figure 17: A switch graph originating a filtration problem.

Given a model over the switch frame generated by S it is clear that no two points will be identified since $(n, 0n)' \vDash' \neg \Diamond \Diamond \top$ and $(n, 0m)' \vDash' \Diamond \Diamond \top$ for $n \neq m$.

We were also unable to adapt the game-based argument to establish the PSPACE upper bound for the satisfiability problem of $\mathcal{H}(@)$, see [3, 7]. The problem is, when adding a new world to a particular component, how to update the other components without falling in a infinite loop of verifications.

The study of the logics corresponding to carefully chosen subclasses may result in more treatable problems and may lead to a better understanding of the general case. One interesting example is the class of switch graphs representing that each edge can only be crossed a certain number of times, modelling the finite durability of certain resources, e.g. roads, bridges, product stocks, etc. The class of switches required to model this is very simple (see Example 1.2), which may be reflected positively in an easier understanding of certain properties of the originating logics, e.g. the finite model property.

Relation with Reactive Logics There are no obvious connections (e.g. translations) between these switch logics and the ones studied in [19]. Even if we can extract a

reactive frame from a switch graph by considering its local behaviour (its paths without jumps), the $\Diamond_P$ operator allows a global access over all the reactive states of a point that cannot be mimicked by the operators considered here. The two approaches are two transversal extensions of classical modal logic with different expressivity. Applications may assert which should be developed or even determine the necessity of joining both views.

Further Extensions Many other hybrid operators have been studied and it would be interesting to investigate how they would play in this context. A particularly natural extension of these logics would be to enrich $\mathcal{H}_s(@)$ with computational tree logic's (CTL) operators ([10]) or even μ-calculus ([31]). This would greatly reinforce our ability to reason about the behaviour of a switch graph and increase its usability. The way this would be done is not completely clear and it would depend on one's particular interests.

If we want to reason about k agents acting in the graph (in Section 2.3 there are such examples) and they are not allowed to jump, we are not interested in the whole set of sequences of edges, but only in the ones that the agents may cross, which are determined by their position at each moment. To express these restrictions, it may be necessary to include in the language the possibility of referring explicitly to the distribution of the agents in the graph. In this direction, a possibility would be to model each agent's location by variables, maybe even introduce another sort of variables to deal with it like we did with nominals. In the used language we would be able to express for example that each agent can be only in one point at a time:

$$@_i(a \wedge \neg j) \rightarrow @_j \neg a,$$

considering a to be an agent variable. Also, to express their movement, one could associate a specific modal operator $\boxtimes_a$ with each agent, and say:

$$@_i(a \wedge \neg j \wedge \Diamond j) \rightarrow \boxtimes_a(j \rightarrow (@_j a \wedge @_i \neg a)).$$

References

[1] C. Areces. *Logic Engineering. The Case of Description and Hybrid Logics*. PhD thesis, Institute for Logic, Language and Computation, University of Amsterdam, Amsterdam, The Netherlands, October 2000.

[2] C. Areces, P. Blackburn, and S. R. Delany. Bringing them all together. *Journal of Logic and Computation*, 11:657–669, 2001.

[3] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for Hybrid logics. In *Proceedings of the 13th International Workshop and 8th Annual Conference of the EACSL on Computer Science Logic*, CSL '99, pages 307–321, London, UK, 1999. Springer-Verlag.

[4] C. Areces, F. Carreiro, S. Figueira, and S. Mera. Basic model theory for memory logics. In *WoLLIC*, pages 20–34, 2011.

[5] H. Barringer and D. M. Gabbay. Modal and temporal argumentation networks. In *Essays in Memory of Amir Pnueli*, pages 1–25, 2010.

[6] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *LOGIC JOURNAL OF IGPL*, 8(3):339–365, 2000.

[7] P. Blackburn, J. F. A. K. v. Benthem, and F. Wolter. *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA, 2006.

[8] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[9] P. Blackburn and B. ten Cate. Pure extensions, proof rules, and Hybrid axiomatics. *Studia Logica*, 84(2):277–322, 2006.

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

[11] R. A. Bull. An approach to tense logic. *Theoria*, 36(3):282–300, 1970.

[12] M. Crochemore and D. M. Gabbay. Reactive automata. *Information and Computation*, In Press, Accepted Manuscript:–, 2011.

[13] D. M. Gabbay. A Theory of Hypermodal Logics: Mode Shifting in Modal Logic. *Journal of Philosophical Logic*, 31(3):211–243, June 2002.

[14] D. M. Gabbay. Reactive Kripke semantics and arc accessibility. In D. F. M. P. Carnielli, W., editor, *Proceedings of CombLog 2004*, pages 7–20. Centre for Logic and Computation, University of Lisbon, 2004.

[15] D. M. Gabbay. Introducing reactive Kripke semantics and arc accessibility. In *Pillars of Computer Science*, pages 292–341, 2008.

[16] D. M. Gabbay. Reactive Kripke Models and Contrary to Duty Obligations. In R. van der Meyden and L. van der Torre, editors, *DEON*, volume 5076 of *Lecture Notes in Computer Science*, pages 155–173. Springer, 2008.

[17] D. M. Gabbay. Reactive intuitionistic tableaux. *Synthese*, pages 1–17, 2010. 10.1007/s11229-010-9781-8.

[18] D. M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*, volume 148 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2003.

[19] D. M. Gabbay and S. Marcelino. Modal Logics of Reactive Frames. *Studia Logica*, 93(2-3):405–446, 2009.

[20] D. M. Gabbay and K. Schlechta. Defeasible inheritance systems and reactive diagrams. *Logic Journal of the IGPL*, 17(1):1–54, 2009.

[21] D. M. Gabbay and K. Schlechta. Reactive Preferential Structures and Nonmonotonic Consequence. *Review of Symbolic Logic*, 2(2):414–450, 2009.

[22] D. M. Gabbay and K. Schlechta. An Analysis of Defeasible Inheritance Systems. In *Logical Tools for Handling Change in Agent-Based Systems*, Cognitive Technologies, pages 251–293. Springer Berlin Heidelberg, 2010.

[23] D. M. Gabbay and K. Schlechta. A theory of hierarchical consequence and conditionals. *J. of Logic, Lang. and Inf.*, 19:3–32, January 2010.

[24] G. Gargov and V. Goranko. Modal logic with names. *Journal of Philosophical Logic*, 22:607–636, 1993. 10.1007/BF01054038.

[25] D. Harel and A. Pnueli. *On the development of reactive systems*, pages 477–498. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[26] M. Huth and M. Ryan. Logic in computer science: Modelling and reasoning about systems, 1999.

[27] S. Marcelino. *Modal Logic for Changing Systems*. PhD thesis, King's College London, 2011.

[28] S. Modgil. Reasoning about preferences in argumentation frameworks. *Artif. Intell.*, 173:901–934, June 2009.

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

[30] A. N. Prior. Modality and quantification in S5. *The Journal of Symbolic Logic*, Vol. 21(No. 1):pp. 60–62, (Mar., 1956).

[31] U. Sattler and M. Vardi. The hybrid μ-calculus. In *Proceedings of IJCAR'01*, Siena, June 2001.

[32] J. van Benthem. An essay on sabotage and obstruction. In D. Hutter and W. Stephan, editors, *Mechanizing Mathematical Reasoning*, volume 2605 of *Lecture Notes in Computer Science*, pages 268–276. Springer, 2005.

[33] J. van Benthem. Dynamic logic for belief revision. *Journal of Applied Non-classical Logics*, 17:129–155, 2007.

[34] A. Zanardo. Branching-Time Logic with Quantification over Branches: The Point of View of Modal Logic. *The Journal of Symbolic Logic*, 61(1):pp. 1–39, 1996.