A logic for n-dimensional hierarchical refinement

Alexandre Madeira
HASLab - INESC TEC & Univ. Minho, Braga, Portugal
amadeira@inesctec.pt

Manuel A. Martins
CIDMA - Dep. Mathematics, Univ. Aveiro, Portugal
martins@ua.pt

Luís S. Barbosa
HASLab - INESC TEC & Univ. Minho Braga, Portugal
lsb@di.uminho.pt

Hierarchical transition systems provide a popular mathematical structure to represent state-based software applications in which different layers of abstraction are represented by inter-related state machines. The decomposition of high level states into inner sub-states, and of their transitions into inner sub-transitions is common refinement procedure adopted in a number of specification formalisms.

This paper introduces a hybrid modal logic for k-layered transition systems, its first-order standard translation, a notion of bisimulation, and a modal invariance result. Layered and hierarchical notions of refinement are also discussed in this setting.

1 Motivation and aims

Figure 1 depicts a high level behavioural model of a strongbox controller in the form of a transition system with three states. The strongbox can be open, closed, or going through an authentication process. The model can be formalised in some sort of modal logic, so that state transitions can be expressed, possibly combined with hybrid features to refers to specific, individual states. Recall that the qualifier *hybrid* [1] applies to extensions of modal languages with symbols, called *nominals*, which explicitly refer to individual states in the underlying Kripke frame. A satisfaction operator $@_i\varphi$ is included standing for φ holding in the state named by nominal i. For example, in propositional hybrid logic [2] and assuming a set of nominals $\mathrm{Nom} = \{closed, get\ access, open\}$, we can express the dynamics depicted in the diagram of Figure 1, *e.g.*,

- that the state *get access* is accessible from the state *closed*, with $@_{closed}\Diamond get\ access$, or
- that the state *open* is not directly accessible from *closed*, with $\Diamond open \rightarrow \neg closed$.

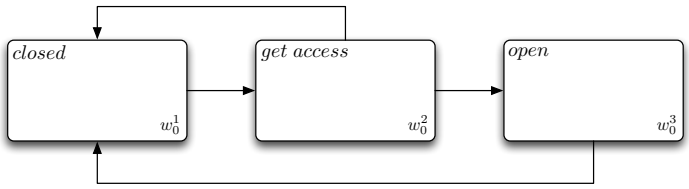

Figure 1: An abstract strongbox behavioural model.

This high level vision of the strongbox controller can be refined by decomposing not only its internal states, but also its transitions. Thus, each 'high-level' state gives rise to a new, local transition system, and each 'high-level'-transition is decomposed into a number of 'intrusive' transitions from sub-states of the 'down level'-transition system corresponding to the refinement of the original source state, to sub-states of the corresponding refinements of original target states. For instance, the (upper) *close* state can

Eerke Boiten, John Derrick & Steve Reeves (Eds.):
Refinement Workshop 2015 (Refine '15)
EPTCS 209, 2016, pp. 40–56, doi:10.4204/EPTCS.209.4

be refined into a (inner) transition system with two (sub) states, one, *idle*, representing the system waiting for the order to proceed for the *get access* state and, another one, *blocked*, capturing a system which is unable to proceed with the opening process (e.g. when authorised access for a given user was definitively denied). In this scenario, the upper level transition from *closed* to *get access* can be realised by, at least, one intrusive transition between the *closed* sub-state *idle* and the *get access* sub-state *identification* where the user identification to proceed is supposed to be checked. Figure 2 illustrates the result of this refinement step.

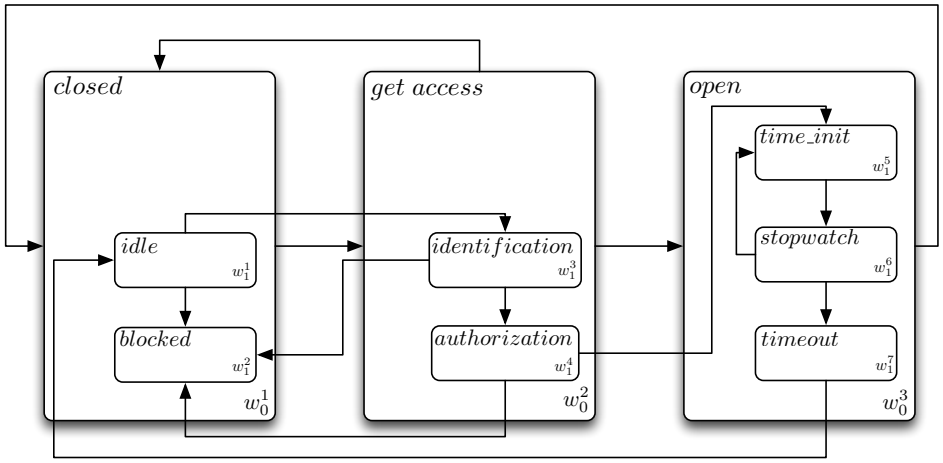

Figure 2: A 2-layered refined strongbox model.

Still the specifier may go even further. For example, he may like to refine the *get access* sub-state *authorisation* into the more fine-grained transition structure depicted in Figure 3. This third-level view includes a sub-state corresponding to each one of the possible three attempts of password validation, as well as an auxiliary state to represent the authentication success.

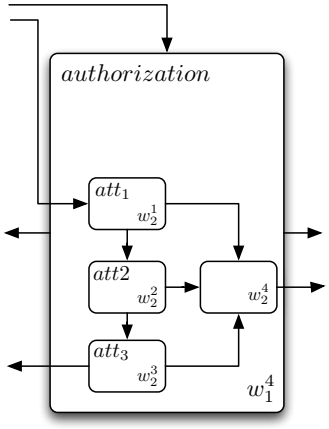

Figure 3: Fragment of the 3-layered refined strongbox model.

Such an hierarchical way to design a system is quite natural and somehow inherent to well known design formalisms such as David Harel's statecharts [8] and the subsequent UML hierarchical state-machines [7], and action refinement [5], among others.

This paper introduces a hierarchical hybrid logic in order to express, and reason about, requirements which typically involve transitions between designated states in different local transition systems, such as, for example, the ones designated by *identification* and *blocked* in Figure 2. This extends our previous work [13] on hierarchical logic in order to capture truly intrusive transitions which are required to express complex software designs as described *e.g.* with statecharts. Suitable notions of bisimulation, and corresponding invariance results, as well as layered and hierarchical refinement are introduced and illustrated.

The paper is organised as follows: Section 2 introduces the logic, whose basic modal theory, including a standard translation to first-order logic and a modal invariance result, is discussed in section 3. Layered and hierarchical refinements are considered in section 4. Finally, section 5 concludes the paper and points out some current developments.

2 The logic

This section introduces a multi-layer hybrid logic to reason upon hierarchical transition systems. The adoption of the term 'hybrid' is addressed to the terminology of the modal logic community (eg. [2, 1]). The 'hybrid' nature of the formalism is regarded to the combination of aspects of first-order and modal logic. Note that this should not be confused with the different, usual, meaning of the the same term to mention systems with mixed continuous and discrete behaviours.

We start fixing a few notational conventions. Given a family $A = (A_i)_{i\in\{0,\cdots,n\}}$, we denote by $A[k]$ the sub-family $A[k] = (A_i)_{i\in\{0,\cdots,k\}}$. Given a predicate $P \subseteq S_1 \times \cdots \times S_n$ we denote by $P|_k$ the restriction of P to its first k components, i.e. the predicate $P|_k \subseteq S_1 \times \cdots \times S_k$ such that

$$P|_k = \big\{(s_1,\cdots,s_k) | P(s_1,\cdots,s_k,s_{k+1},\cdots,s_n) \text{ for some } s_r \in S_r, r \in \{k+1,\cdots,n\}\big\}$$

Given a relation $R \subseteq (S_1 \times \cdots \times S_n)^2$, we denote by $R|_k$ the relation $R|_k \subseteq (S_1 \times \cdots \times S_k)^2$ such that

$$R|_k = \big\{(s_1,\cdots,s_k,s'_1,\cdots,s'_k) | R(s_1,\cdots,s_k,s_{k+1},\cdots,s_n,s'_1,\cdots s'_k,s'_{k+1},\cdots s'_n) \text{ for } s_r, s'_r \in S_r, r \in \{k+1,\cdots,n\}\big\}$$

The logic can now be introduced as follows.

Signatures

Signatures are n-families of disjoint, possible empty, sets of symbols

$$\Delta^n = \big(\mathrm{Prop}_k, \mathrm{Nom}_k\big)_{k\in\{0,\cdots,n\}}$$

Example 2.1 *To express the strongbox model introduced above as a running example, we have to define a signature Δ^2 for the three layers presented. Note that, for sake of simplicity, the level-subscripts of nominals are omitted in the diagrams above.* 0*-level symbols consist of the set of nominals* $\mathrm{Nom}_0 = \{closed_0, get\_access_0, open_0\}$*, and a set of propositions* Prop_0 *including, for instance, a proposition*

$safe\_state_0$ to be assigned to the 0*-states where the machine is not opened. For the* 1*-level signature we consider a set of nominals*

$$\mathrm{Nom}_1 = \{idle_1, blocked_1, identification_1, authorization_1, time\_init_1, stopwatch_1, time\_out_1\}$$

and a set of propositions Prop_1 *which may include, for example, a proposition $timed\_state_1$ to be assigned to timed dependent* 1*-states (e.g. the inner states of the one named by $open_0$ and the 1-state named by $authorization_1$). The fragment presented in Figure 3 entails the inclusion of nominals att_{12}, att_{22} and att_{32} in* Nom_2.

Formulas

The set of formulas $Fm(\Delta^n)$ is the n-family recursively defined, for each k, by

$$\varphi_0 \ni i_0 \mid p_0 \mid \neg\varphi_0 \mid \varphi_0 \wedge \varphi_0 \mid @_{i_0}\varphi_0 \mid \diamond_0 \varphi_0$$

$$\varphi_0^b \ni i_0 \mid p_0 \mid @_{i_0}\varphi_0 \mid \diamond_0 \varphi_0$$

and

$$\varphi_k \ni \varphi_{k-1}^b \mid i_k \mid p_k \mid \neg\varphi_k \mid \varphi_k \wedge \varphi_k \mid @_{i_k}\varphi_k \mid \diamond_k \varphi_k$$

where for any $k \in \{1,\ldots,n\}$, the basic formulas are defined by

$$\varphi_{k-1}^b \ni i_{k-1} \mid p_{k-1} \mid \varphi_{k-2}^b \mid @_k\varphi_{k-1} \mid \diamond_{k-1} \varphi_{k-1}$$

for $k \in \{2,\cdots,n\}$, $p_k \in \mathrm{Prop}_k$ and $i_k \in \mathrm{Nom}_k$.

For each $k \in \{0,\ldots,n\}$, the strict k-layered formulas $SFm(k,\Delta^n)$ are defined as the fragments of $Fm(\Delta^n)$ given by the grammar

$$\varphi_k \ni p_k \mid i_k \mid \neg\varphi_k \mid \varphi_k \wedge \varphi_k \mid @_{i_k}\varphi_k \mid \diamond_k \varphi_k$$

The positive fragments of $Fm(\Delta^n)$ and $SFm(k,\Delta^n)$, i.e., the sets of sentences built by the corresponding grammars but excluding negations, are denoted by $Fm^+(\Delta^n)$ and $SFm^+(k,\Delta^n)$, respectively.

Example 2.2 *This language is able to express properties of very different natures, some of them easily identifiable in our running example. For instance, we may express inner-outer relations between named states (e.g. $@_{idle_1} closed_0$ or $@_{att_{12}} open_0$) as well as a variety of transitions. Those include, for example, the layered transition $@_{get\_access_0} \diamond_0 open_0$, the* 0*-internal transition $@_{identification_1} \diamond_1 authorisation_1$ or the* 0*-intrusive transitions $@_{idle_1} \diamond_1 authorisation_1$ and $get\_access_0 \rightarrow \diamond_1 open_0$. Example 2.3 provides further examples of the logic expressiveness.*

Models

Definition 2.1 (n-layered models) *A n-layered model $M \in \mathrm{Mod}^n(\Delta^n)$ is a tuple*

$$M = (W^n, D^n, R^n, V^n)$$

recursively defined as follows:

- $W^n = (W_k)_{k\in\{0,\cdots,n\}}$ *is a family of disjoint sets*

- $D^n \subseteq W_0 \times \cdots \times W_n$ *is a predicate such that, denoting by* D_k *the k-restriction* $D^n|_k$*, for each* $k \in \{0,\cdots,n\}$*, verifies*

$$W_k = \{v_k | D_k(w_0,\cdots,w_{k-1},v_k), \textit{ for some } w_0,\cdots,w_{k-1} \textit{ such that } D_{k-1}(w_0,\cdots,w_{k-1})\}$$

- $R^n = \big(R_k \subseteq D_k \times D_k\big)_{k\in\{0,\cdots,n\}}$ *is a n-family of binary relations;*
- $V^n = (V_k^{\text{Prop}}, V_k^{\text{Nom}})_{k\in\{0,\cdots,n\}}$ *is a family of pairs of functions*
  - $V_0^{\text{Prop}} : \text{Prop}_0 \to \mathscr{P}(W_0)$ *and* $V_k^{\text{Prop}} : \text{Prop}_k \times D_{k-1} \to \mathscr{P}(W_k)$ *for any* $k > 0$*; and*
  - $V_k^{\text{Nom}} : \text{Nom}_k \to W_k$.

For each $k \in \{0,\cdots n\}$, model $M_k = (W^n[k], D^n|_k, R^n[k], V^n[k])$, is said the k-restriction of $M = \text{Mod}^n(\Delta^n)$.

A specific, particularly well-behaved class of layered models, very important in refinement situation, is defined as follows:

Definition 2.2 (Hierarchical Model) *A n-layered model* $M = (W^n, D^n, R^n, V^n) \in \text{Mod}^n(\Delta^n)$ *is said to be* hierarchical *if for any* $k \in \{1,\cdots,n\}$ $R_k|_{k-1} = R_{k-1}$.

Example 2.3 *Our running example is clearly a hierarchical model. Examples of non-hierarchical layered models can be achieved by removing some* 0*-transitions depicted in Figure 2 (e.g. the one linking the named states* $closed_0$ *and* get_access_0*). Observe that, in this case, one has* $@_{closed_0}\Diamond_1 get\_access_0$ *but* $\neg@_{closed_0}\Diamond_0 get\_access_0$.

Satisfaction

Let M be a n-layered model. The satisfaction consists of a family of relations $\models^n = (\models_k)_{k\in\{0,\cdots,n\}}$ defined, for each $w_r \in W^r$, $r \in \{0,\cdots,k\}$, $k \leq n$, such that $D_k(w_0,\cdots w_k)$, as follows:

- $M_k, w_0,\cdots,w_k \models_k \varphi^b_{k-1}$ iff $M_{k-1}, w_0,\cdots,w_{k-1} \models_{k-1} \varphi^b_{k-1}$
- $M_k, w_0,\cdots,w_k \models_k p_k$ iff $w_k \in V_k^{\text{Prop}}(p_k, w_0,\cdots,w_{k-1})$
- $M_k, w_0,\cdots,w_k \models_k i_k$ iff $w_k = V_k^{\text{Nom}}(i_k)$ and $D_k(w_0,\cdots,w_{k-1}, V_k^{\text{Nom}}(i_k))$
- $M_k, w_0,\cdots,w_k \models_k \varphi_k \wedge \varphi'_k$ iff $M_k, w_0,\cdots,w_k \models_k \varphi_k$ and $M_k, w_0,\cdots,w_k \models_k \varphi'_k$
- $M_k, w_0,\cdots,w_k \models_k \neg\varphi_k$ iff it is false that $M_k, w_0,\cdots,w_k \models_k \varphi_k$
- $M_k, w_0,\cdots,w_k \models_k @_{i_k}\varphi_k$ iff $M_k, w_0,\cdots w_{k-1}, V_k^{\text{Nom}}(i_k) \models_k \varphi_k$ and $D_k(w_0,\cdots w_{k-1}, V_k^{\text{Nom}}(i_k))$
- $M_k, w_0,\cdots,w_k \models_k \Diamond_k\varphi_k$ iff $M, v_0,\cdots,v_k \models_k \varphi_k$ for some $v_r \in W_r$, $r \in \{0,\cdots,k\}$, such that $(w_0,\ldots,w_k)R_k(v_0,\cdots,v_k)$.

Example 2.4 *Let us illustrate this notion of satisfaction verifying the validity of* $@_{idle_1} closed_0$ *in state* w_0^1 *of model* M *in the running example. For arbitrary* w_1, w_2*,*

$$\begin{array}{ll} & M_2, w_0, w_1, w_2 \models_2 @_{idle_1} closed_0 \\ \Leftrightarrow & \quad \{ \textit{ defn. of } \models_2 \} \\ & M_1, w_0^1, w_1 \models_1 @_{idle_1} closed_0 \\ \Leftrightarrow & \quad \{ \textit{ defn. of } \models_1 \} \end{array}$$

$$
\begin{array}{ll}
 & M_1, w_0^1, V_1^{\mathrm{Nom}}(idle_1) \models_1 closed_0, \text{ and } D_2(w_0^1, V_1^{\mathrm{Nom}}(idle_1)) = D_2(w_0^1, w_1^1) \\
\Leftrightarrow & \quad \{ \textit{ defn. of } \models_1 \} \\
 & V_0^{\mathrm{Nom}}(closed_0) = w_0^1 \text{ and } D_2(V_0^{\mathrm{Nom}}(closed_0), w_1^1) = D_2(w_0^1, w_1^1)
\end{array}
$$

As a second illustration consider,

$$
\begin{array}{ll}
 & M_1, w_0^2, w_1^3 \models_1 \Diamond_1 get\_access_0 \wedge \neg \Diamond_0 get\_access_0 \\
\Leftrightarrow & \quad \{ \textit{ defn. of } \models_1 \} \\
 & M_1, w_0^2, w_1^3 \models_1 \Diamond_1 get\_access_0 \text{ and it is false that } M_1, w_0^2, w_1^3 \models_1 \Diamond_0 get\_access_0 \\
\Leftrightarrow & \quad \{ \textit{ defn. of } \models_1 \} \\
 & \text{there are } v_0, v_1 \text{ such that } (w_0^2, w_1^3) R_1 (v_0, v_1) \text{ and } M_1, v_0, v_1 \models_1 get\_access_0 \\
 & \text{and} \\
 & \text{it is false that there is a } r_0 \text{ such that } (w_0^2) R_0 (r_0) \text{ and } M_0, r_0 \models_0 get\_access_0 \\
\Leftrightarrow & \quad \{ \textit{ defn. of } \models_1 \} \\
 & \text{there are } v_0, v_1 \text{ such that } (w_0^2, w_1^3) R_1 (v_0, v_1) \text{ and } v_0 = V_0^{\mathrm{Nom}}(warning_0) \\
 & \text{(when } v_0 = w_0^2 \text{ and } v_1 = w_1^4 \text{) and} \\
 & \text{there is not a } r_0 \text{ such that } (w_0^2) R_0 (r_0) \text{ and } r_0 = V_0^{\mathrm{Nom}}(warning_0)
\end{array}
$$

3 Basic modal theory

This section discusses three basic ingredients in a modal theory: the existence of a standard translation to first-order logic, a notion of bisimulation and a modal invariance result.

3.1 Standard translation

Beyond the theoretical interest of this characterization, a standard translation to first-order logic paves the way to the use of a number of tools to provide assistance and effective support for the refinement strategies suggested here.

Signature translation: An n-layered signature $\Delta^n = (\mathrm{Nom}_n, \mathrm{Prop}_n)$ induces, for each $k \in \{0, \ldots, n\}$, a first-order signature (S^k, F^k, P^k) as follows:

- $S^k = \{S_0, \cdots, S_k\}$;
- F^k is the (S^{k^*}, S^k)-family of function symbols consisting of:
  - for each $r \in \{0, \cdots, k\}$, $F^k_{\rightarrow S_r} = \{i_r | i_r \in \mathrm{Nom}_r\}$
  - and $F_{\omega \rightarrow S} = \emptyset$ for the other cases.
- P^k is a S^{k^*}-family of predicate symbols such that for any $r \in \{0, \cdots, k\}$:
  - $P_{S_0, \cdots S_r} = \{D_r\}$
  - $P_{S_1 \cdots S_r} = \mathrm{Prop}_r$
  - $P_{S_0, \cdots S_r, S_0, \cdots S_r} = \{R_r\}$
  - and $P_\omega = \emptyset$ for the other cases.

Models translation: Let M be a Δ^n model. For each $k \in \{0,\cdots,n\}$, the (S^k,F^k,P^k)-model M_k^*, corresponding to the translation of M_k, is built as follows. For each $r \in \{0,\cdots,k\}$:

- $M^*_{k\,S_r} = W_r$
- for each $i_r :\rightarrow S_r$, $M^*_{k\,i_r} = V_r^{\mathrm{Nom}}(i_r)$
- for any $p_r \in P_{S_r}$, $M^*_{k\,p_r}(w_0,\cdots w_r) = V_r^{\mathrm{Prop}}(p_r,w_0,\cdots,w_r)$
- $M^*_{k\,D_r} = D_r$
- $M^*_{k\,R_r} = R_r$

Sentences translation: The translation of sentences is recursively defined as follows:

$$\begin{array}{rcll}
\mathrm{ST}^k_{x_0,\cdots,x_k}(\varphi^b_{k-1}) & = & \mathrm{ST}^{k-1}_{x_0,\cdots,x_{k-1}}(\varphi^b_{k-1}) & \\
\mathrm{ST}^k_{x_0,\cdots,x_k}(p_k) & = & p_k(x_0,\cdots,x_k) & p_k \in \mathrm{Prop}_k \\
\mathrm{ST}^k_{x_0,\cdots,x_k}(i_k) & = & i_k = x_k & i_k \in \mathrm{Nom}_k \\
\mathrm{ST}^k_{x_0,\cdots,x_k}(@_{i_k}(\varphi_k)) & = & D_k(x_0,\cdots,x_{k-1},i_k) \wedge \mathrm{ST}^k_{x_0,\cdots,x_{k-1},i_k}(\varphi_k) & i_k \in \mathrm{Nom}_k \\
\mathrm{ST}^k_{x_0,\cdots,x_k}(\diamond_k\varphi_k) & = & (\exists y_0,\cdots y_k)\big(D_k(y_0,\cdots,y_k) \wedge & \\
 & & R_k(x_0,\cdots,x_k,y_0,\ldots,y_k) \wedge \mathrm{ST}^k_{y_0,\ldots y_k}(\varphi_k)\big) & \\
\mathrm{ST}^k_{x_0,\cdots,x_k}(\varphi_k \wedge \varphi'_k) & = & \mathrm{ST}^k_{x_0,\cdots,x_k}(\varphi_k) \wedge \mathrm{ST}^k_{x_0,\cdots,x_k}(\varphi'_k) & \\
\mathrm{ST}^k_{x_0,\cdots,x_k}(\neg\varphi_k) & = & \neg\mathrm{ST}^k_{x_0,\cdots,x_k}(\varphi_k) &
\end{array}$$

Example 3.1

$$\begin{array}{rl}
\mathrm{ST}^1_{x_0,x_1}(@_{idle_1}closed_0) & = D_1(x_0,idle_1) \wedge \mathrm{ST}^1_{x_0,idle_1}(closed_0) \\
 & = D_1(x_0,idle_1) \wedge \mathrm{ST}^0_{x_0}(closed_0) \\
 & = D_1(x_0,idle_1) \wedge (closed_0 = x_0)
\end{array}$$

Theorem 3.1 *Let M be a n-layered model of Δ^n and φ_k, $k \in \{0,\cdots,n\}$, a formula of $Fm_k(\Delta^n)$. Then,*

$$M_k,w_0,\cdots,w_k \models_k \varphi_k \ \textit{iff}\ \overline{M_k} \models \mathrm{ST}^k_{x_0\cdots x_k}(\varphi_k)$$

where $\overline{M_k}$ is the $x_0,\cdots,x_k$-expansion of M_k^* such that $\overline{M_k}_{x_r} = w_r$, for any $r \in \{0,\cdots,k\}$, and $\models$ stands for the first order satisfaction relation.

Proof. The proof is done by induction over the sentences and satisfaction structure. For $k = 0$, the theorem boils down to the corresponding result for usual standard translation for propositional hybrid logic (see e.g. [2]). For the remaining cases:

Case of formulas φ^b_{k-1}

$$\begin{array}{rl}
 & M_k,w_0,\cdots,w_k \models_k \varphi^b_{k-1} \\
\Leftrightarrow & \quad \{\ \text{defn. of } \models_k\} \\
 & M_{k-1},w_0,\cdots,w_{k-1} \models_{k-1} \varphi^b_{k-1}
\end{array}$$

$$
\begin{array}{ll}
\Leftrightarrow & \{\text{ I.H.}\} \\
& \overline{M_{k-1}} \models \mathrm{ST}^{k-1}_{x_0,\cdots,x_{k-1}}(\varphi^b_{k-1}) \\
\Leftrightarrow & \{\text{ defn. of } \mathrm{ST}^k \text{ and } \mathrm{ST}^{k-1}_{x_0,\cdots,x_{k-1}}(\varphi^b_{k-1}) \text{ does not depend on } x_k\} \\
& \overline{M_k} \models \mathrm{ST}^k_{x_0,\cdots,x_k}(\varphi^b_{k-1})
\end{array}
$$

Case of formulas $@_{i_k}\varphi$

$$
\begin{array}{ll}
& M_k, w_0, \cdots, w_k \models_k @_{i_k}\varphi \\
\Leftrightarrow & \{\text{ defn. of } \models_k\} \\
& M_k, w_0, \ldots, w_{k-1}, V^{\mathrm{Nom}}_k(i_k) \models_k \varphi \text{ and } D_k(w_0, \cdots, w_{k-1}, V^{\mathrm{Nom}}_k(i_k)) \\
\Leftrightarrow & \{\text{ I.H. + defn. of } \overline{M_k}\} \\
& \overline{M_k}_{D_k}(\overline{M_k}_{x_0}, \cdots, \overline{M_k}_{x_{k-1}}, \overline{M_k}_{i_k}) \text{ and } \overline{M_k} \models \mathrm{ST}^k_{x_0,\cdots,x_{k-1}}(\varphi) \\
\Leftrightarrow & \{\models \text{defn}\} \\
& \overline{M_k} \models D_k(x_0, \cdots, x_{k-1}, i_k) \wedge \mathrm{ST}^k_{x_0,\cdots,x_{k-1},i_k}(\varphi) \\
\Leftrightarrow & \{\mathrm{ST}^k \text{ defn}\} \\
& \overline{M_k} \models \mathrm{ST}^k_{x_0,\cdots,x_k}(@_{i_k}\varphi)
\end{array}
$$

Case of formulas i_k

$$
\begin{array}{ll}
& M_k, w_0, \cdots, w_k \models_k i_k \\
\Leftrightarrow & \{\text{ defn. of } \models_k\} \\
& w_k \in V^{\mathrm{Nom}}_k(i_k) \text{ and } D_k(w_0, \cdots, w_{k-1}, V^{\mathrm{Nom}}_k(i_k)) \\
\Leftrightarrow & \{\text{ defn. of } \models\} \\
& M_{k i_k} = w_k \text{ and } M_{D_k}(w_0, \cdots, w_{k-1}, M_{k i_k}) \\
\Leftrightarrow & \{\text{ defn. of } \overline{M_k}\} \\
& \overline{M_{k i_k}} = \overline{M_k}_{x_k} \text{ and } \overline{M}_{D_k}(\overline{M}_{x_0}, \cdots, \overline{M}_{x_{k-1}}, \overline{M_k}_{i_k}) \\
\Leftrightarrow & \{\text{ defn. of } \models\} \\
& \overline{M_k} \models i_k = x_k \wedge D_k(x_0, \cdots, x_{k-1}, i_k) \\
\Leftrightarrow & \{\text{ defn. of } \mathrm{ST}^k\} \\
& \overline{M_k} \models \mathrm{ST}^k_{x_0,\cdots,x_k}(i_k)
\end{array}
$$

Case of formulas p_k

$$
\begin{array}{ll}
& M_k, w_0, \cdots, w_k \models_k p_k \\
\Leftrightarrow & \{\text{ defn. of } \models_k\} \\
& w_k \in V^{\mathrm{Prop}}_k(p_k, w_0, \cdots, w_{k-1}) \\
\Leftrightarrow & \{\text{ defn. of } M_k\} \\
& M_{k p_k}(w_0, \cdots, w_k)
\end{array}
$$

$$\begin{array}{ll} \Leftrightarrow & \{ \text{ defn. of } \overline{M_k} \text{ + defn. } \models \} \\ & \overline{M_k} \models p_k(x_0, \cdots, x_k) \\ \Leftrightarrow & \{ \text{ defn. of ST}^k \} \\ & \overline{M_k} \models \mathrm{ST}^k_{x_0,\cdots,x_k}(p_k) \end{array}$$

Case of formulas $\diamond_k \varphi$

$$\begin{array}{ll} & M_k, w_0, \cdots, w_k \models_k \diamond_k \varphi \\ \Leftrightarrow & \{ \text{ defn. of } \models_k \} \\ & M_k, v_0, \ldots, v_k \models_k \varphi \text{ for some } v_r \in W_r, \text{ such that } (w_0, \ldots, w_k) R_k (v_0, \ldots, v_k) \\ \Leftrightarrow & \{ \text{ defn. of } \overline{M_k} \text{ + FOL semantics + } R_k \subseteq D_k \times D_k \} \\ & \overline{M_k} \models (\exists y_0, \cdots y_k) \big(D(x_0, \cdots, x_{k-1}, y_k, \cdots, y_n) \wedge R_k(x_0, \cdots, x_k, y_0 \cdots, y_k) \wedge \mathrm{ST}^k_{y_0,\cdots,y_k}(\varphi) \big) \\ \Leftrightarrow & \{ \text{ defn. of ST} \} \\ & \overline{M_k} \models \mathrm{ST}^k_{x_0,\cdots,x_k}(\diamond_k \varphi) \end{array}$$

The proof for the boolean connectives is straightforward. ■

3.2 Bisimulation and modal invariance

Bisimulation is the main conceptual tool to compare transition systems. The originality in the definition below is the way the layered structured is taken into account in the *zig-zag* conditions. This is illustrated in Figure 4 below. The remaining components are, as expected, completely standard in hybrid logic. Note, for example, the condition imposed on nominals which makes bisimilarity a quite fine-grained equivalence. Back to Figure 4 this condition forces us not to consider nominals in the transition structures represented there.

Definition 3.1 (n-layered bisimulation) *Let M and M' be two n-layered models over the signature $\Delta^n = (\mathrm{Prop}_k, \mathrm{Nom}_k)_{k \in \{0,\cdots,n\}}$. A family of relations*

$$B = (B_k \subseteq D_k \times D'_k)_{k \in \{0,\cdots,n\}}$$

is a n-layered bisimulation (n-bisimulation for short) if, for any $k \in \{0, \cdots, n\}$, whenever

$$(w_0, \cdots, w_k) B_k (w'_0, \cdots, w'_k)$$

we have that:

($ATOM_k$)
1. *for each $p_k \in \mathrm{Prop}_k$,*
   - i. *if $k = 0$ we have $w_0 \in V_0^{\mathrm{Prop}}(p_0)$ iff $w'_0 \in V'^{\mathrm{Nom}}_0(p_0)$;*
   - ii. *otherwise, $w_k \in V_k(p_k, w_0, \cdots, w_{k-1})$ iff $w'_k \in V'_k(p_k, w'_0, \cdots, w'_{k-1})$.*
2. *for each $i_k \in \mathrm{Nom}_k$,*
   - i. *if k=0, $(V_0^{\mathrm{Nom}}(i_0)) B_k (V'^{\mathrm{Nom}}_0(i_0))$;*
     otherwise, $(w_0, \cdots, w_{k-1}, V_k^{\mathrm{Nom}}(i_k)) B_k (w'_0, \cdots, w'_{k-1}, V'^{\mathrm{Nom}}_k(i_k))$,
   - ii. *$w_k = V_k(i_k)$ and $D_k(w_0, \cdots, w_{k-1}, V_k^{\mathrm{Nom}}(i_k))$ iff $w'_k = V'_k(i_k)$ and $D'_k(w'_0, \cdots, w'_{k-1}, V'^{\mathrm{Nom}}_k(i_k))$*

(*ZIG_k*) *for any* $v_r \in W_r$, $r \in \{0,\cdots,k\}$, *such that* $(w_0,\cdots,w_k)R_k(v_0,\cdots,v_k)$, *there are* $v'_r \in W'_r$, $r \in \{0,\cdots,k\}$, *such that* $(w'_0,\cdots,w'_k)R'_k(v'_0,\cdots v'_k)$ *and*

$$(v_0,\cdots,v_k)B_k(v'_0,\cdots,v'_k) \tag{1}$$

(*ZAG_k*) *for any* $v'_r \in W'_r$, $r \in \{0,\cdots,k\}$, *such that* $(w'_0,\cdots,w'_k)R'_k(v'_0,\cdots,v'_k)$, *there are* $v_r \in W_r$, $r \in \{0,\cdots,k\}$, *such that* $(w_0,\cdots,w_k)R_k(v_0,\cdots,v_k)$ *satisfy (1)*

Lemma 3.1 *In the conditions of the previous definition,* $B[k] \subseteq D_k \times D_k$ *is a k-bisimulation between* M_k *and* M'_k, *for any* $k \in \{0,\cdots,n\}$.

Theorem 3.2 *Let M and* M' *be two n-layered models over the signature* Δ^n *and B a n-layered bisimulation. Then, for* $k \in \{0,\cdots,n\}$ *and* $w_k \in W_k$, $w'_k \in W'_k$, *such that* $(w_0,\cdots,w_k)B_k(w'_0,\cdots,w'_k)$ *and any* $\varphi_k \in SFm(k,\Delta^n)$,

$$M_k, w_0,\cdots,w_k \models_k \varphi_k \text{ iff } M'_k, w'_0,\cdots,w'_k \models_k \varphi_k$$

Proof. The proof is done by induction over the sentences and satisfaction structure. For $k = 0$, the theorem boils down to the corresponding modal invariance result for propositional hybrid logic (see e.g. [2]). For the remaining cases:

Case of formulas $\diamond_k\varphi_k$

$$\begin{array}{ll} & M_k, w_0,\cdots,w_k \models_k \diamond_k\varphi_k \\ \Leftrightarrow & \quad \{ \text{ defn. of } \models_k \} \\ & M_k, v_0,\ldots,v_k \models_k \varphi_k \text{ for some } v_r \in W_r, r \in \{0,\cdots,k\} \\ & \text{ such that } (w_0,\cdots,w_n)R_k(v_0,\cdots,v_k) \\ \Leftrightarrow & \quad \{ \text{ Step } (\star) \} \\ & M'_k, v'_0,\cdots,v'_k \models_k \varphi_k \text{ for some } v'_r \in W'_r, r \in \{0,\cdots,k\} \\ & \text{ such that } (w'_0,\cdots,w'_n)R'_k(v'_0,\cdots,v'_k) \\ \Leftrightarrow & \quad \{ \text{ defn. of } \models_k \} \\ & M'_k, w'_0,\cdots,w'_k \models_k \diamond_k\varphi_k \end{array}$$

Step $(\star)$**:** For the up-down implication, the existence of $v'_0,\cdots,v'_k$ such that $(w_0,\cdots,w_k)B_k(v_0,\cdots,v_k)$ is assured by the recursive application of (ZIG_k). Then conclude by I.H. The down-up implication is proved similarly, but resorting to the (ZAG_k) condition.

Case of formulas $@_{i_k}\varphi_k$

$$\begin{array}{ll} & M_k, w_0,\cdots,w_k \models_k @_{i_k}\varphi_k \\ \Leftrightarrow & \quad \{ \text{ defn. of } \models_k \} \\ & M_k, w_0,\cdots,w_{k-1}, V_k^{\text{Nom}}(i_k) \models_k \varphi_k \text{ and } D_k(w_0,\cdots,w_{k-1},V_k^{\text{Nom}}(i_k)) \\ \Leftrightarrow & \quad \{ \ (w_0,\cdots,w_k)B_k(w'_0,\cdots,w'_k) + ATOM_k \text{ (2.i.) + I.H.} \} \\ & M'_k, w'_0,\cdots,w'_{k-1}, V'^{\text{Nom}}_k(i_k) \models_k \varphi_k \text{ and } D'_k(w'_0,\cdots,w'_{k-1},V'^{\text{Nom}}_k(i_k)) \\ \Leftrightarrow & \quad \{ \text{ defn. of } \models_k \} \\ & M'_k, w'_0,\cdots,w'_k \models_k @_{i_k}\varphi_k \end{array}$$

Case of formulas i_k

$$
\begin{array}{ll}
 & M_k, w_0, \cdots, w_k \models_k i_k \\
\Leftrightarrow & \quad \{ \text{ defn. of } \models_k \} \\
 & w_k = V_k^{\text{Nom}}(i_k) \text{ and } D_k(w_0, \cdots, w_{k-1}, V_k^{\text{Nom}}(i_k)) \\
\Leftrightarrow & \quad \{ ATOM_k \text{ (2.i.)} \} \\
 & w'_k = V'^{\text{Nom}}_k(i_k) \text{ and } D'_k(w'_0, \cdots, w'_{k-1}, V'^{\text{Nom}}_k(i_k)) \\
\Leftrightarrow & \quad \{ \text{ defn. of } \models_k \} \\
 & M'_k, w'_0, \cdots, w'_k \models_k i_k
\end{array}
$$

The proof for propositions is analogous and for the boolean connectives is straightforward.

■

Definition 3.1 boils down to the following version of bisimulation for hierarchical systems:

Definition 3.2 (n-hierarchical bisimulation) *Let M and M' be two n-hierarchical models over $\Delta^n = (\text{Prop}_k, \text{Nom}_k)_{k\in\{0,\cdots,n\}}$. A* n-hierarchical bisimulation *consists of a relation $B \subseteq D_n \times D'_n$ such that, for any $k \in \{0, \cdots, n\}$ and for any $w_r \in W_r$ and $w'_r \in W'_r$, $r \in \{0, \ldots, k\}$ such that $(w_0, \cdots, w_n)B(w'_0, \cdots, w'_n)$, we have that:*

($ATOM$) *1. for each $p_k \in \text{Prop}_k$,*
- *i. if $k = 0$ we have $w_0 \in V_0^{\text{Prop}}(p_0)$ iff $w'_0 \in V'^{\text{Nom}}_0(p_0)$;*
- *ii. otherwise, $w_k \in V_k(p_k, w_0, \cdots, w_{k-1})$ iff $w'_k \in V'_k(p_k, w'_0, \cdots, w'_{k-1})$*

2. for each $i_k \in \text{Nom}_k$,
- *i. if k=0, $(V_0^{\text{Nom}}(i_0))B|_0(V'^{\text{Nom}}_0(i_0))$;*
  otherwise, $(w_0, \cdots, w_{k-1}, V_k^{\text{Nom}}(i_k))B|_k(w'_0, \cdots, w'_{k-1}, V'^{\text{Nom}}_k(i_k))$,
- *ii. $w_k = V_k(i_k)$ and $D_k(w_0, \cdots, w_{k-1}, V_k^{\text{Nom}}(i_k))$ iff $w'_k = V'_k(i_k)$ and $D'_k(w'_0, \cdots, w'_{k-1}, V'^{\text{Nom}}_k(i_k))$*

(ZIG) *for any $v_r \in W_r$, $r \in \{0, \cdots, n\}$, such that $(w_0, \cdots, w_n)R(v_0, \cdots, v_n)$, there are $v'_r \in W'_r$, $r \in \{0, \cdots, n\}$, such that $(w'_0, \cdots, w'_n)R'(v'_0, \cdots v'_n)$ and*

$$(v_0, \cdots, v_n)B(v'_0, \cdots, v'_n) \tag{2}$$

(ZAG) *for any $v'_r \in W'_r$, $r \in \{0, \ldots, n\}$, such that $(w'_0, \cdots, w'_n)R'(v'_0, \cdots, v'_n)$, there are $v_r \in W_r$, $r \in \{0, \cdots, n\}$, such that $(w_0, \cdots, w_n)R(v_0, \cdots, v_n)$ satisfying (2)*

Corollary 3.1 *Let M and M' be two n-hierarchical models over the signature Δ^n and B an n-hierarchical bisimulation. Then for any $w_k \in W_k$ and $w'_k \in W'_k$, $k \in \{0, \cdots, n\}$ such that $(w_0, \cdots, w_n)B(w'_0, \cdots, w'_n)$ and for any $\varphi \in Fm_n(\Delta^n)$,*

$$M, w_0, \cdots, w_n \models_n \varphi \text{ iff } M', w'_0, \cdots, w'_n \models_n \varphi$$

Proof. First of all observe that a n-hierarchical bisimulation $B \subseteq D_n \times D'_n$ induces a n-layered bisimulation $\overline{B}$ by taking, for each $k \in \{0, \cdots, n\}$, the component $\overline{B}_k = B|_k$.

Since $\varphi \in Fm(\Delta^n)$ then $\varphi \in Fm_n(\Delta^n)$. Hence since $M, w_0, \ldots, w_n \models_n \varphi$ iff $M_k, w_0, \cdots, w_k \models_k \varphi$. Moreover, since M and M' are hierarchical, we have that $(w_0, \cdots, w_k)B[k](w'_0, \cdots, w'_k)$. By Theorem 3.2 to achieve at $M'_k, w'_0, \cdots, w'_k \models_k \varphi$. Hence $M', w'_0, \cdots, w'_n \models_n \varphi$. ■

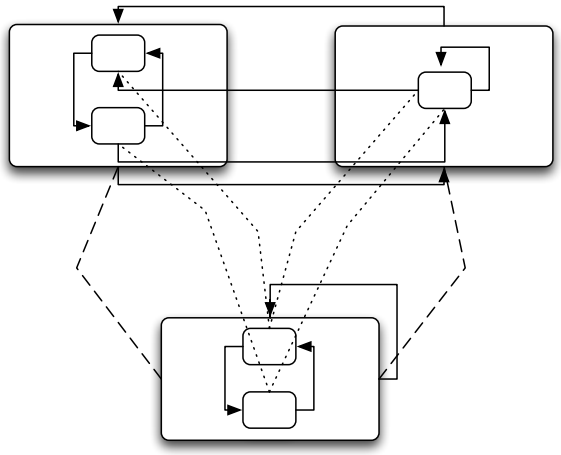

Figure 4: A 2-hierarchical bisimulation.

4 Refinement

4.1 Simulation

As usual, simulation entails a notion of refinement. However, as done with bisimulations in section 3, a distinction is made between the general notion of n-layered simulation and more well-behaved' *n-hierarchical* one. Definitions and results are as expected. Thus,

Definition 4.1 (n-layered simulation) *Let M and M' be two n-layered models over the signature $\Delta^n = (\mathrm{Prop}_k, \mathrm{Nom}_k)_{k\in\{0,\cdots,n\}}$. A family of relations*

$$S = (S_k \subseteq S_k \times S'_k)_{k\in\{0,\cdots,n\}}$$

is a n-layered simulation from M to M' if for any $k \in \{0,\cdots,n\}$ and for any $w_r \in W_r$ and $w'_r \in W'_r$, $r \in \{0,\cdots,k\}$, whenever $(w_0,\cdots,w_k)S_k(w'_0,\cdots,w'_k)$, we have that:

($ATOM_k$)
1. *for each $p_k \in \mathrm{Prop}_k$,*
   - *i. when $k=0$, if $w_0 \in V_0^{\mathrm{Prop}}(p_0)$ then $w'_0 \in V'^{\mathrm{Nom}}_0(p_0)$;*
   - *ii. otherwise, $w_k \in V_k(p_k, w_0,\cdots,w_{k-1})$ implies that $w'_k \in V'_k(p_k, w'_0,\cdots,w'_{k-1})$*
2. *for each $i_k \in \mathrm{Nom}_k$,*
   - *i. if k=0, $(V_0^{\mathrm{Nom}}(i_0))S_k(V'^{\mathrm{Nom}}_0(i_0))$;*
     otherwise, $(w_0,\cdots,w_{k-1},V_k^{\mathrm{Nom}}(i_k))S_k(w'_0,\cdots,w'_{k-1},V'^{\mathrm{Nom}}_k(i_k))$,
   - *ii. $w_k = V_k(i_k)$ and $D_k(w_0,\cdots,w_{k-1},V_k^{\mathrm{Nom}}(i_k))$ implies that $w'_k = V'_k(i_k)$ and $D'_k(w'_0,\cdots,w'_{k-1},V'^{\mathrm{Nom}}_k(i_k))$*

(ZIG_k) *for any $v_r \in W_r$, $r \in \{0,\cdots,k\}$, such that $(w_0,\cdots,w_k)S_k(v_0,\cdots,v_k)$, there are $v'_r \in W'_r$, $r \in \{0,\cdots,k\}$, such that $(w'_0,\cdots,w'_k)R'_k(v'_0,\cdots v'_k)$ and $(v_0,\cdots,v_k)S_k(v'_0,\cdots,v'_k)$*

Theorem 4.1 *Let M and M' be n-layered models over the signature Δ^n and S an n-layered simulation from M to M'. Then for any $w_k \in W_k$ and $w'_k \in W'_k$, $k \in \{0,\cdots,n\}$ such that $(w_0,\cdots,w_k)S_k(w'_0,\cdots,w'_k)$ and for any $\varphi_k \in SFm^+(k,\Delta^n)$,*

$$M_k, w_0,\cdots,w_k \models_k \varphi_k \text{ implies that } M'_k, w'_0,\cdots,w'_k \models_k \varphi_k$$

Proof. Straightforward from Theorem 3.2. ■

The definition specialises to one for n-hierarchical systems.

Definition 4.2 (n-hierarchical simulation) *Let M, M' be two n-hierarchical models over $\Delta^n = (\mathrm{Prop}_n, \mathrm{Nom}_n)$. A n-*hierarchical simulation *consists of a relation $S \subseteq D_n \times D'_n$ such that, for any $k \in \{0,\cdots,n\}$ and for any $w_r \in W_r$ and $w'_r \in W'^r$, $r \in \{0,\ldots,k\}$, whenever $(w_0,\cdots,w_n)S(w'_0,\cdots,w'_n)$, we have that*

(*ATOM*) 1. *for each $p_k \in \mathrm{Prop}_k$,*
   i. *when $k=0$, if $w_0 \in V_0^{\mathrm{Prop}}(p_0)$ then $w'_0 \in V'^{\mathrm{Nom}}_0(p_0)$;*
   ii. *otherwise, $w_k \in V_k(p_k, w_0,\cdots,w_{k-1})$ implies that $w'_k \in V'_k(p_k, w'_0,\cdots,w'_{k-1})$*

2. *for each $i_k \in \mathrm{Nom}_k$,*
   i. *if k=0, $(V_0^{\mathrm{Nom}}(i_0))S|_0(V'^{\mathrm{Nom}}_0(i_0))$;*
   otherwise, $(w_0,\cdots,w_{k-1},V_k^{\mathrm{Nom}}(i_k))S|_k(w'_0,\cdots,w'_{k-1},V'^{\mathrm{Nom}}_k(i_k))$,
   ii. *$w_k = V_k(i_k)$ and $D_k(w_0,\cdots,w_{k-1},V_k^{\mathrm{Nom}}(i_k))$ implies that $w'_k = V'_k(i_k)$ and $D'_k(w'_0,\cdots,w'_{k-1},V'^{\mathrm{Nom}}_k(i_k))$*

(*ZIG*) *for any $v_r \in W_r$, $r \in \{0,\cdots,n\}$, such that $(w_0,\ldots,w_n)S(v_0,\cdots,v_n)$, there are $v'_r \in W'_r$, $r \in \{0,\cdots,n\}$, such that $(w'_0,\cdots,w'_n)R'(v'_0,\cdots v'_n)$ and $(v_0,\ldots,v_n)S(v'_0,\ldots,v'_n)$*

Given two n-layered models M and M', we say that M' l-simulates M, in symbols $M \rightharpoonup_l M'$ if there exists a total horizontal n-layered simulation S from M to M'. Analogously, given two n-hierarchical models M and M', we say that M' h-simulates M, in symbols $M \rightharpoonup_h M'$, if there exists a total n-hierarchical simulation S from M to M'.

Example 4.1 *Back to our running example, suppose now that, to meet an additional safety requirement, it is imposed that, whenever blocked, the strongbox has to be reset under some specific administrative permissions. This scenario is depicted in Figure 5. Clearly, this updated model hierarchically simulates*

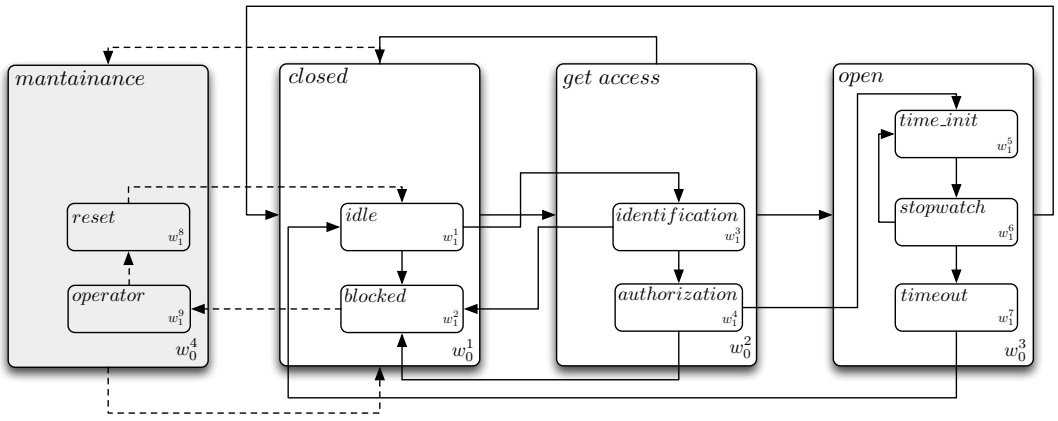

Figure 5: Strongbox with administrative reset.

the original one in Figure 2. Actually the former model is a sub-model of the latter.

Corollary 4.1 *Let M and M' be two n-hierarchical models over the signature Δ^n and S an n-hierarchical simulation. Then for $k \in \{0,\cdots,n\}$, $w_k \in W_k$ and $w'_k \in W'_k$, such that $(w_0,\cdots,w_n)S(w'_0,\cdots,w'_n)$ and for any $\varphi \in Fm_n^+(\Delta^n)$,*

$$M,w_0,\cdots,w_n \models_n \varphi \text{ implies that } M',w'_0,\cdots,w'_n \models_n \varphi$$

Proof. Straightforward from Theorem 4.1. ■

4.2 Refinement

Finally, we have all ingredients to define refinement, distinguishing again the general n-layered case from the n-hierarchical one.

Definition 4.3 (Layered refinement) *Let Δ^{n+k} be a $n+k$-layered signature and $\Delta^n = \Delta^{n+k}[n]$. Let $M = (W^n,D^n,R^n,V^n) \in \mathrm{Mod}^n(\Delta^n)$ and $N = (W'^{n+k},D'^{n+k},R'^{n+k},V'^{n+k}) \in \mathrm{Mod}^{n+k}(\Delta^{n+k})$. We say that N is an* layered refinement *of M, in symbols $M \rightsquigarrow_l N$, whenever $M \rightharpoonup_l N_n$.*

Definition 4.4 (Hierarchical refinement) *Let Δ^{n+k} be a $n+k$-hierarchical signature and $\Delta^n = \Delta^{n+k}[n]$. Let $M = (W^n,D^n,R^n,V^n) \in \mathrm{Mod}^n(\Delta^n)$ and $N = (W'^{n+k},D'^{n+k},R'^{n+k},V'^{n+k}) \in \mathrm{Mod}^{n+k}(\Delta^{n+k})$. We say that N is an* hierarchical refinement *of M, in symbols $M \rightsquigarrow_h N$, whenever $M \rightharpoonup_h N_n$.*

Theorem 4.2 *Let $M = (W^n,D^n,R^n,V^n)$ and $N = (W'^{n+k},D'^{n+k},R'^{n+k},V'^{n+k})$ two layered models of Δ^n and Δ^{n+k} respectively. Then, if $M \rightsquigarrow_l N$, we have for any $\varphi \in SFm^+(k,\Delta^n)$ and for any $w_r \in W'_r$, $r \in \{n+1,\cdots,n+k\}$,*

$$M,w_0,\cdots,w_n \models_n \varphi \text{ implies that } N,w_0,\cdots,w_n,\cdots,w_{n+k} \models_{n+k} \varphi$$

Proof. Straightforward from Corollary 4.1. ■

Theorem 4.3 *Let $M = (W^n,D^n,R^n,V^n)$ and $N = (W'^{n+k},D'^{n+k},R'^{n+k},V'^{n+k})$ two hierarchical models of Δ^n and Δ^{n+k} respectively. Then, if $M \rightsquigarrow_h N$, we have for any $\varphi \in Fm^+(\Delta^n)$ and for any $w_r \in W'_r$, $r \in \{n+1,\cdots,n+k\}$,*

$$M,w_0,\cdots,w_n \models_n \varphi \text{ implies that } N,w_0,\cdots,w_n,\cdots,w_{n+k} \models_{n+k} \varphi$$

Proof. Straightforward from Theorem 4.1. ■

Example 4.2 *It is easy to show that the model considered in Example 4.1 is a* 2*-hierarchical refinement of the one presented in Figure 2. Actually, its* 1*-level restriction simulates the model presented in Figure 1, as illustrated in Figure 6.*

Finally, Figure 7 illustrates the associated stepwise refinement process in which simulation steps combine with refinements until reaching a suitable implementation. Note that the strictly vertical arrows correspond to hierarchical steps along which, up to given level, the original and the refined transition systems are bisimilar. The diagonal arrows represent proper simulations between them.

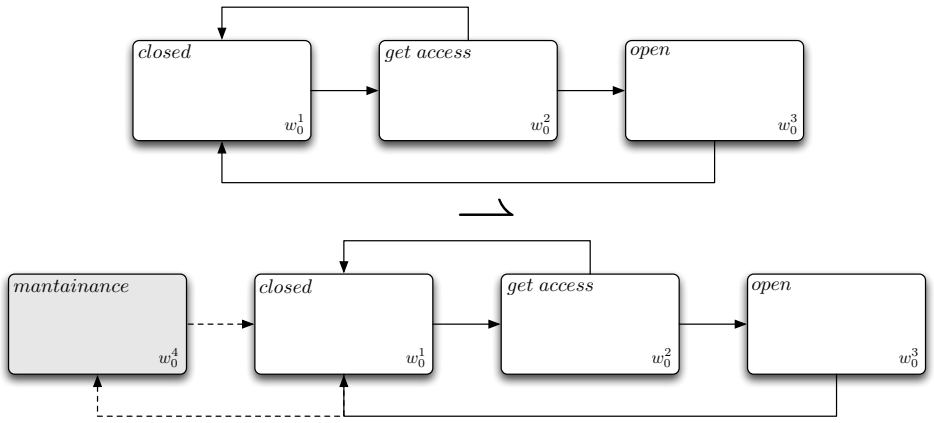

Figure 6: An hierarchical refinement.

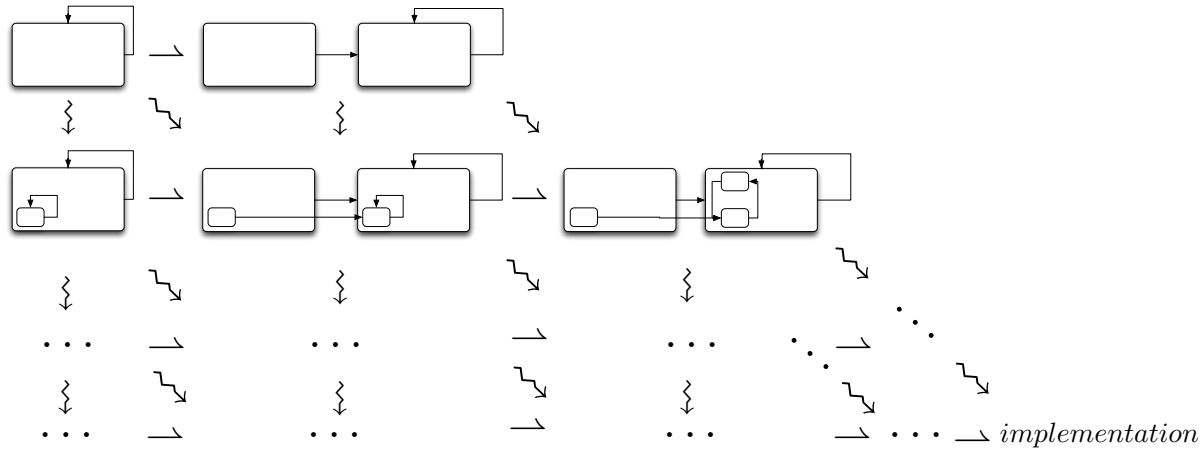

Figure 7: A (hierarchical) stepwise refinement process.

5 Conclusions and further work

The paper introduced a hybrid modal logic for reasoning about k-layered transition systems and support horizontal and hierarchical refinement. The logic is expressive enough to capture different forms of intra- and inter-level transitions present in most formalisms used in software specification and analysis with an hierarchical flavour, spanning from D. Harel's *statecharts* [8] to the mobile *ambients* [3] of A. Gordon and L. Cardelli, Actually this work is rooted on the authors' previous study of what was called *hybrid hierarchical logic*, $\mathscr{H}\mathscr{H}\mathscr{L}$, in reference [13] and, although being much more restrictive in the sort of expressible transitions, represented a first step in characterising a logic for hierarchical structures. Indeed $\mathscr{H}\mathscr{H}\mathscr{L}$ arises from building a extra hybrid level (with new sets of nominals and modalities) on top of standard, propositional hybrid logic. This process, in full generality, is called *hybridisation* [11, 14, 4] and consists of taking an arbitrary logic, framed as an institution [6] and systematically developing on top of it the syntax and semantic features of hybrid logic. Refinement in hybridised logics was studied by the authors in [12].

The development of suitable notions of both horizontal and hierarchical refinements is one of the paper's contributions. Current work is therefore mainly concerned with proof-of-concept applications, namely the study of variants of k-layered logics devoted to specific approaches in software engineering design. For instance, in a recent institutional rendering of UML [10], the formalisation of UML state-machines leaves out hierarchical states (see [9]), a limitation that may be addressed in our framework. Other future research directions are concerned with decidability, the development of a calculus and proof support.

Acknowledgements. This work is funded by ERDF - European Regional Development Fund, through the COMPETE Programme, and by National Funds through FCT within project PTDC/EEI-CTP/4836/2014. A. Madeira is supported by the FCT grant SFRH/BPD/103004/2014. Finally, M. Martins is supported by FCT project UID/MAT/04106/2013 at CIDMA and the EU FP7 Marie Curie PIRSES-GA-2012-318986 project GeT-Fun: Generalizing Truth-Functionality.

References

[1] Patrick Blackburn (2000): *Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto*. *Logic Journal of IGPL* 8(3), pp. 339–365. Available at `http://dx.doi.org/10.1093/jigpal/8.3.339`.

[2] Torben Brauner (2010): *Hybrid Logic and its Proof-Theory*. Applied Logic Series, Springer.

[3] Luca Cardelli & Andrew D. Gordon (1998): *Mobile Ambients*. In Maurice Nivat, editor: *Foundations of Software Science and Computation Structure, First International Conference, FoSSaCS'98, Lisbon, Portugal, March 28 - April 4, 1998, Proceedings*, *Lecture Notes in Computer Science* 1378, Springer, pp. 140–155. Available at `http://dx.doi.org/10.1007/BFb0053547`.

[4] Razvan Diaconescu & Alexandre Madeira (2015): *Encoding hybridized institutions into first-order logic*. *Mathematical Structures in Computer Science* FirstView, pp. 1–44, doi:10.1017/S0960129514000383. Available at `http://journals.cambridge.org/article_S0960129514000383`.

[5] Rob J. van Glabbeek & Ursula Goltz (2001): *Refinement of actions and equivalence notions for concurrent systems*. *Acta Inf.* 37(4/5), pp. 229–327. Available at `http://link.springer.de/link/service/journals/00236/bibs/1037004/10370229.htm`. Available at `http://dx.doi.org/10.1007/s002360000041`.

[6] Joseph A. Goguen & Rod M. Burstall (1992): *Institutions: Abstract Model Theory for Specification and Programming*. *J. ACM* 39(1), pp. 95–146. Available at `http://doi.acm.org/10.1145/147508.147524`.

[7] O. M. G. Group: *UML Specification, Version 2.0*.

[8] David Harel (1987): *Statecharts: A Visual Formalism for Complex Systems*. *Sci. Comput. Program.* 8(3), pp. 231–274. Available at `http://dx.doi.org/10.1016/0167-6423(87)90035-9`.

[9] Alexander Knapp, Till Mossakowski & Markus Roggenbach (2014): *An Institutional Framework for Heterogeneous Formal Development in UML*. *CoRR* abs/1403.7747. Available at `http://arxiv.org/abs/1403.7747`.

[10] Alexander Knapp, Till Mossakowski & Markus Roggenbach (2015): *Towards an Institutional Framework for Heterogeneous Formal Development in UML - - A Position Paper -*. In Rocco De Nicola & Rolf Hennicker, editors: *Software, Services, and Systems - Essays Dedicated to Martin Wirsing on the Occasion of His Retirement from the Chair of Programming and Software Engineering*, *Lecture Notes in Computer Science* 8950, Springer, pp. 215–230, doi:10.1007/978-3-319-15545-6-15.

[11] Alexandre Madeira (2013): *Foundations and techniques for software reconfigurability*. Ph.D. thesis, Universidades do Minho, Aveiro and Porto (Joint MAP-i Doctoral Programme).

[12] Alexandre Madeira, Manuel A. Martins, Luís Soares Barbosa & Rolf Hennicker (2015): *Refinement in hybridised institutions*. *Formal Asp. Comput.* 27(2), pp. 375–395, doi:10.1007/s00165-014-0327-6.

[13] Alexandre Madeira, Renato Neves, Manuel Martins & Luis Barbosa (2014): *Introducing Hierarchical Hybrid Logic*. In: *Advances in Modal Logic 2014*, pp. 74 – 78.

[14] Manuel A. Martins, Alexandre Madeira, Răzvan Diaconescu & Luís Soares Barbosa (2011): *Hybridization of Institutions*. In A. Corradini, B. Klin & C. Cîrstea, editors: *Algebra and Coalgebra in Computer Science (CALCO 2011, Winchester, UK, August 30 - September 2, 2011)*, *Lecture Notes in Computer Science* 6859, Springer, pp. 283–297. Available at `http://dx.doi.org/10.1007/978-3-642-22944-2_20`.