IfI Technical Reports

Institute of Computer Science,
Clausthal University of Technology

IfI-05-09

Clausthal-Zellerfeld 2005

Expressing and Verifying Temporal and Structural Properties of Mobile Agents

Marek A. Bednarczyk[1], Wojciech Jamroga[2] and Wiesław Pawłowski[1]
[1] Institute of Computer Science, Polish Academy of Sciences, Gdańsk, Poland
{m.bednarczyk,W.Pawlowski}@ipipan.gda.pl
[2] Institute of Computer Science, Clausthal University of Technology, Germany
wjamroga@in.tu-clausthal.de

Abstract

The paper deals with logics for expressing temporal and structural properties of *Petri hypernets*, a visual formalism for modeling mobile agents. In particular, we consider how such logics can be build as a composition of two formalisms—one for expressing the temporal, another for expressing the structural properties of multi-agent systems. The problem of model checking properties of a class of composed logics on Petri hypernets is shown to be PSPACE-complete.

Keywords: multi-agent systems, mobile agents, Petri hypernets, temporal logics, model checking.

1 Introduction

Petri hypernets is a formalism that has been proposed recently for modeling mobile agents [3]. Due to its Petri net heritage it can be considered as a *visual* framework. The structure of each individual agent is represented as a single Petri net. In a hypernet, which is a collection of such agent nets, some agent nets can be contained in other nets. This leads to a hierarchy with some agents controlling other agents. Agents within a hypernet may act asynchronously or synchronize their actions. Evolution of the hypernet may also, like in the case of mobile ambients [4], change their position within the overall hierarchy of agents.

Here, we propose a language for reasoning about dynamics of the structure of systems representable within that framework. The language combines two families of modal operators—one family to cope with the temporal, the other to deal with the spatial (or structural) dimension. From this perspective, our approach follows [12]. Unlike Franceschet et al., however, we do not start with two logics to cope with each dimension and look what we get from their combination. Instead, we interpret the language of the combination directly in a class of Kripke structures containing Petri hypernets. Only

then we approach the problem of how the resulting logic can be seen as a combination of its two components.

For the purpose of the presentation we have chosen to represent the temporal and the structural properties of agents in the style of Computational Tree Logic CTL. It is a feature of Petri hypernets that the evolving hierarchy of agents always remains a *tree*. This enabled us to strengthen the structural part of the logic by adding "past tense"-like modalities with no computational cost. In our application, that means references to one's super-agents (while the classical "future tense"-like operators address sub-agents of the current agent). As we have already mentioned, the semantics of the logic is given with respect to a class of abstract intermediate models called agent-oriented hyper-transition systems – rather than defined directly over hypernets.

The paper is organized as follows. We begin with a short introduction to Petri hypernets and their basic properties in Section 2, and we demonstrate the ideas on an air traffic modeling example. Then, we discuss some examples of interesting tempo-structural properties that can be specified (and verified) for systems which structure of individuals evolves in time. In Sections 3.4 and 3.5, we present our intermediary models (agent-oriented hyper-transition systems), and define a logic called CTL^2, that we use for describing temporal and structural properties of hypernets. In section 3.6 we explain how CTL^2 can be seen as a combination of two logics. In Section 4, the complexity of model checking for such a combination of logics is studied. Like CTL, also CTL^2 can be model-checked in time linear in the length of the formula and the size of the agent-oriented hypertransition system. Moreover, the model checking problem for CTL^2 turns out to be PSPACE-complete in the length of the formula and the size of the *Petri hypernet* itself. Thus, model checking CTL^2 is no worse than for the logic of mobile ambients or standard temporal logics, which hints that automatic verification of agent properties via Petri hypernets and modal logics like CTL^2 may be feasible as well. Finally, we discuss limitations of the approach presented here and suggest directions for future research.

2 Hypernets — a Visual Formalism for Mobile Agents

The notion of nets, introduced by C. A. Petri in 1962, offers a fundamental model of concurrent computation with spatially distributed components: *places* and *transitions*, see Figure 1. The places are local states in which some *resources* can be stored. Distribution of these resources among the places corresponds to the global state of the net, called *marking*. The transitions can fetch the resources from some places and transport them to other places—thereby changing the global state of the net. The interaction is *static*, *finitary* and *local*, i.e., each transition has a fixed and finite set of places with which it interacts, and these constitute its local environment. The structure of a net is usually represented as a directed bipartite graph, with transitions drawn as boxes, places as circles, and resources as blobs, see monograph [24] for more on Petri nets. Figure 1, for example, presents a net in two states: before and after *firing* a transition.

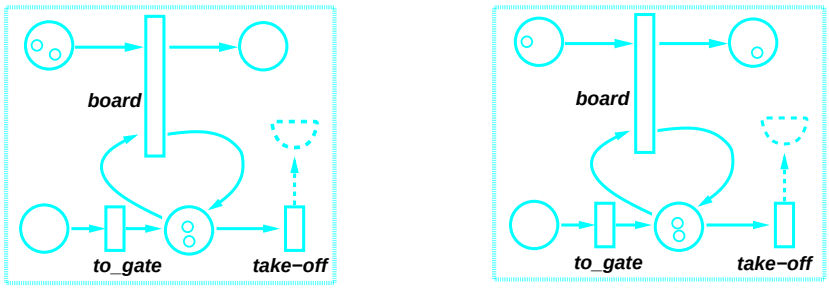

Figure 1: Boarding a plane: before (left) and after (right) event *board*.

The marking represented on the left *enables* transition *board* in several ways. That is, all the *precondition* places of *board* contain resources. Any so enabled transition can *fire*, which amounts to transporting a resource from each precondition place to every postcondition place. The result of firing the transition is depicted on the right of Figure 1.

Since their inception, Petri nets have been generalized in many ways. Initially, people considered the resources to be *facts* (in the case of *elementary* or *C/E nets*), or *quantities*: either natural numbers in the case of discrete *P/T nets*, or non-negative reals in the case of *continuous* nets. In some generalizations, the resources are structured, e.g., they can be elements of some algebra or a data-type.

2.1 Petri Hypernets

In Petri hypernets [3], Petri nets are used to represent *agents*: the structure of each individual agent is modeled with a single net, and agents can occur as tokens in places within other agents. Thus, Petri hypernets (PHN in short) offer a *visual* formalism for modeling distributed and concurrent multi-agent systems. A hypernet $H = \langle \mathcal{N}, m \rangle$ consists of a set of *open nets* and a *hypermarking*. An open net $N \in \mathcal{N}$ consists of a number of modules, represented by sequential Petri nets; among other things, the set of places in N is denoted by P_N, and $P_H = \bigcup_{N \in \mathcal{N}} P_N$ is the set of all places in H. A hypermarking $m : \mathcal{N} \rightharpoonup P_H$ defines the current distribution of nets within other nets.

The idea to use nets as tokens has appeared quite early, and is now known as the *nets-within-nets* approach, see [28]. Intuitively, using nets as tokens leads to a *hierarchy*—a token net is *lower* in the hierarchy than its *owner*, i.e., lower than the net in whose place the "token" net currently sits. In Valk's elaboration, each token net can move between places of its current owner, but there is no proviso for token exchange. Thus, a token net is bound to its owner. Agents organized in this way have a fixed, static hierarchy. Hypernets, in contrast, are capable of dynamically changing that hierarchy. In this respect they are like *mobile ambients* [4].

Any framework which aims to model *individual* agents has to address the problem of

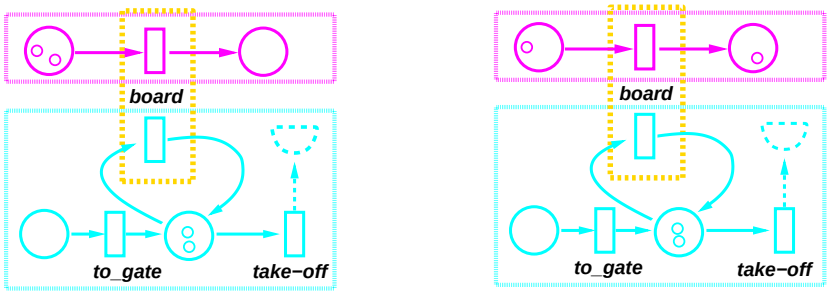

Figure 2: Structured boarding: before (left) and after (right) event *board*.

agent identity. That is, one has to say say what happens to each agent when the multi-agent system evolves. In terms of the nets-within-nets approach this means that one has to say which token nets take part in transition firing. For instance, if the resources presented on the left of Figure 1 correspond to distinguishable agents, one should say *which* of them are chosen for boarding, and where do they move as a result. Now, if transition *board* had one of its postconditions removed, we would either had to put both tokens to the single postcondition place, or face the problem of *agent destruction*. Conversely, if its top precondition was removed together with the tokens, we would have to address the problem of *agent creation*.

In Valk's approach, the problem is avoided by saying that the tokens are not nets, but *references* to nets. This, however, leads to semantic problems when synchronization between an agent and its token nets is considered. In hypernets, to keep the model simple, the problems mentioned above are resolved by simple means. Namely, nets are considered to be synchronized products of *modules*, each module having the structure of a *state machine*. The idea is explained on Figure 2. The net from Figure 1 is split into two components called *modules*. Each transition has *exactly* one precondition and one postcondition in a module. Thus, the problems of identity preservation, creation and destruction of agents are resolved. Cooperation between the modules is enforced by synchronization of transitions with the same names. In this way, a rich structure of behaviors can be modeled, for instance every 1-safe or elementary net can be decomposed as such a synchronous product of sequential state machine components.

2.2 Modeling Mobility with Hypernets

An agent can move within its "owner" by firing appropriate transitions. In order to allow migration of agents from one owner to another, an old concept of message passing is used. The idea is demonstrated in Figure 3, where the structure of two agents, an airport on top, and a plane at the bottom is shown. The airport agent consists of two modules. The bottom one is for handling planes, and the plane agent is indeed located in one of the places. The top module is for handling passengers. The plane agent has

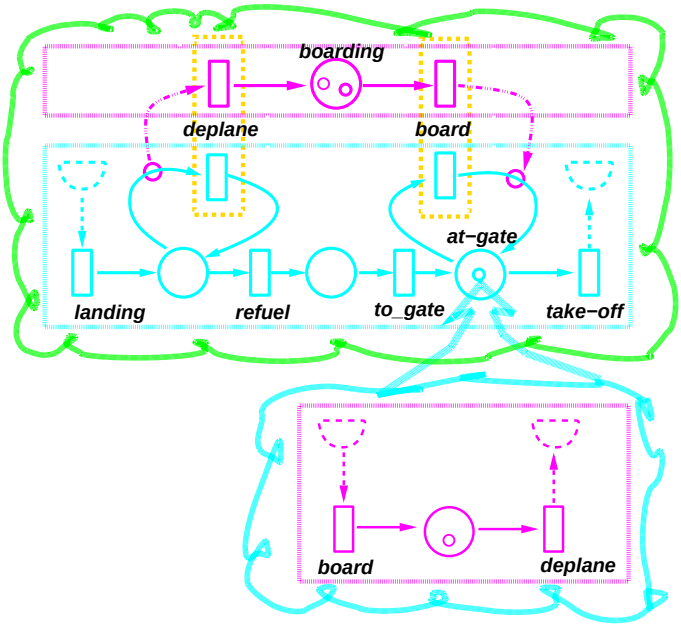

Figure 3: Airport, plane, and 3 passengers

just one module for passenger handling. There are two passengers at the airport, and one in the plane. Any such distribution of agents among other agents' places is called *hypermarking*.

In hypernets, each agent can communicate with its current owner (if any), and with the agents it currently owns. To this end, some of its places are designated as *communication ports*. For instance, the airport agent in Figure 3 is *always* ready to accept a new plane agent from its hypothetical owner by engaging in transition *landing*. Similarly, the plane agent is always prepared to receive a traveller from its owner by engaging in *board* transition. Both agents offer also dual operations for sending planes (resp. travellers), up the hierarchy tree, with transitions *take-off* (resp. *deplane*). The ports for communicating with the owner are usually depicted as dashed semicircles.

Communication with the owner is simple—there is at most one owner. To send a token down, we nominate a module responsible for choosing the addressee agent. In our example, transition *board* in the traveller module of the airport has its output port attached to the output arc from the *board* transition in the plane module of the airport. Thus, the traveller token chosen for boarding in the traveler module will be sent down to the plane token chosen in the plane module for boarding, as shown in Figures 4 and 5.

We assume here that *any* inter-level exchange of agents requires that the agent on the other level is willing to perform the dual action. For instance, the plane agent in Figure 4

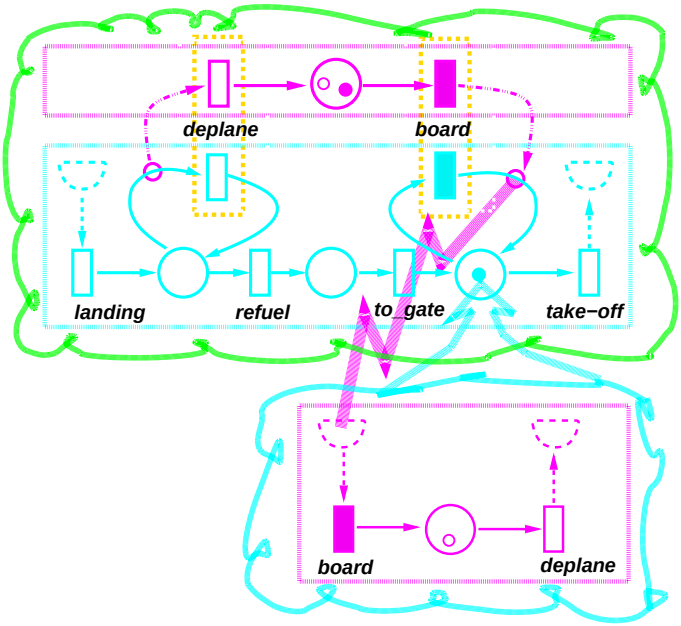

Figure 4: Transaction firing in hypernets: boarding a passenger

is willing to accept travellers, and it is ready for boarding as a token in the airport agent (i.e. it enables transition *board* in the airport module for planes). Also, the *board* transition in the airport traveler module is enabled, and one of the traveler tokens has been chosen for boarding. Boarding a passenger should result in the traveller agent leaving the airport, and entering the plane agent. Formally, the inter-level synchronization takes place. It is facilitated by creation of a temporary communication channel that connects the dual ports in the nets being involved, as indicated in Figure 4. Figure 5 shows the result of boarding. Thus, a single step of the hypernet involves synchronization of 3 transitions called *board* in different modules and levels. Consequently, we call such an atomic step of the hypernet a *transaction*. Also, inter-level matching of ports is required to preserve the *module names*, e.g., the plane agent can only receive an agent from the traveller module of its current owner.

Even though the plane agent on Figure 3 is ready to deplane the traveller agent it owns, *deplane* is not enabled. Simply, the owner of the plane is not ready to engage the plane agent in its *deplane* transition, and take care of the passenger. Thus, the synchronization capabilities of the agents change as the hypernet evolves.

In summary, Petri hypernets offer a hierarchical and modular framework for modeling mobile agents. The global states of a hypernet are hypermarkings, i.e., distributions of agents among places of other agents. It is assumed that the induced ownership rela-

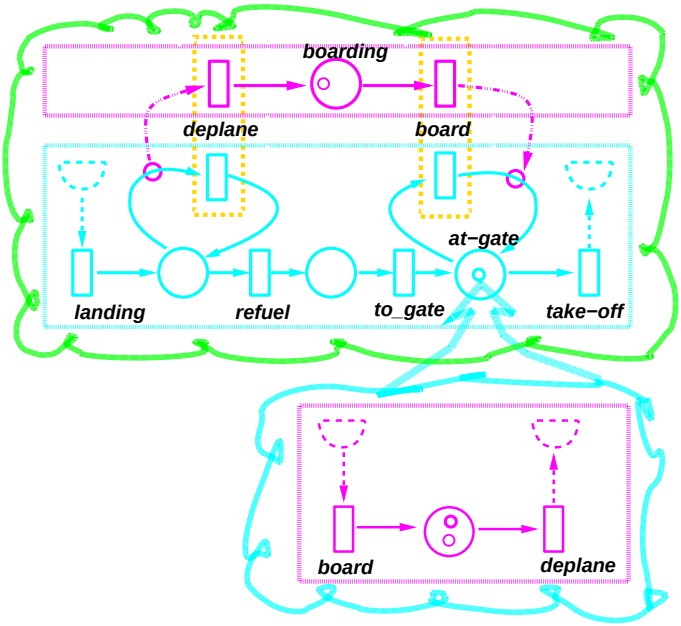

Figure 5: After the transaction

tion is initially a tree, with only one *super-agent* containing all the other agents. Evolution of the hypernet corresponds to firing a transaction of actions, all with the same name. As a result of a transaction, an agent in the hypernet can change its position in the hierarchy. Such a change is possible only under supervision of its immediate owner. Agents stay within modules of the same kind, e.g., a traveller can never enter a plane-handling module of another agent. The main technical result of [3] says that the evolution of hypernets preserves the tree-like character of the hierarchy of agents.

Remark 1 *Let us remark that the number of hypermarkings reachable from the initial one is more than exponential in general. Assume that there are k traveller agents at place* boarding *of the airport agent in Figure 3, and ℓ planes ready to pick them up at place* at-gate*. Then each traveller can board any plane, which gives at least ℓ^k different hypermarkings. The travellers are represented by empty hypernets; the planes also have constant size (cf. Figure 3). Let n be the number of places in the whole hypernet. Taking $k = \ell = n/c$ for a suitable constant c gives us at least $2^{O(n \log n)}$ states in the resulting transition system. Conversely, we can distribute $k + \ell$ agents among their n places in at most $n^{\ell+k}$ ways, which proves $2^{O(n \log n)}$ to be an upper bound too.*

3 Properties of Agents in Petri Hypernets

Hypernets seem to provide a natural formalism for modeling and designing systems of mobile agents. The visual nature of their Petri net heritage has even resulted in a tool with which simple hypernets can be defined and their behaviour simulated. Hypernets will remain, however, only a modeling tool until a language is developed in which one can express properties of agents. Here, we focus on two dimensions of such a specification formalism. The *temporal* dimension captures how the system evolves in time. The *spatial*, or *structural* dimension refers to the agents' position within other agents. From this perspective, our approach resembles [12]. Unlike Franceschet et al., however, we do not start with two logics to cope with each dimension and look what we get from their combination. Instead, we interpret directly the language of the combination in a class of Kripke structures corresponding to Petri hypernets. Only then we approach the problem of how the resulting logic can be seen as a combination of its two components.

A hypermarking in a hypernet associates to each token net a local place of its owner. Here, we neglect these details and keep only the induced information about the subsumption of agents to other agents, i.e., the tree hierarchies of agents. Since the hierarchy change in hypernets is always supervised, we also assume that *subsumption structure* among agents is always a tree.

Remark 2 *In order to make basic statements concerning the two aspects of multi-agent systems, we assume that we deal with* labeled *Petri hypernets, in which agents and places can be labeled with the names of atomic propositions. Namely, we assume that the set* Π *of atomic propositions is partitioned into two disjoint sets:* Π_{ag} *(agent labels) and* Π_{pl} *(place labels),* $\Pi = \Pi_{ag} \cup \Pi_{pl}$*, and that there are two labeling functions given* $\lambda_{ag} : \Pi_{ag} \to \mathcal{P}(\mathcal{N})$ *and* $\lambda_{pl} : \Pi_{pl} \to \mathcal{P}(P_H)$. Agent propositions $p_{ag} \in P_{ag}$ *will hold in the context of each net labeled with* p_{ag} *in* H *(e.g., proposition* airportGD *can be used to designate the Gdańsk airport).* Place propositions $p_{pl} \in P_{pl}$ *will hold in the context of hypermarkings in which every place labeled with* p_{pl} *is inhabited by some token (e.g., proposition* gate7 *can be used to designate the situations in which the gate no. 7 is non-empty). This issue is further formalized in Section 4.1.*

3.1 Temporal Evolution of Systems

There are a number of modal logics that address how a system can evolve in time [9]. The temporal aspect of the system is usually modeled with a *transition system*, in which nodes represent possible situations or *states* of the system, and arcs or *transitions* show how states can change in a single step. In the case of a PHN, states can be naturally thought of as hypermarkings; to determine the outgoing transitions for state q representing hypermarking m, we take arcs to all the states that represent hypermarkings that can be obtained from m by firing a single transaction. Atomic propositions are used to define "instantly observable" properties of states. In a model, the valuation of propo-

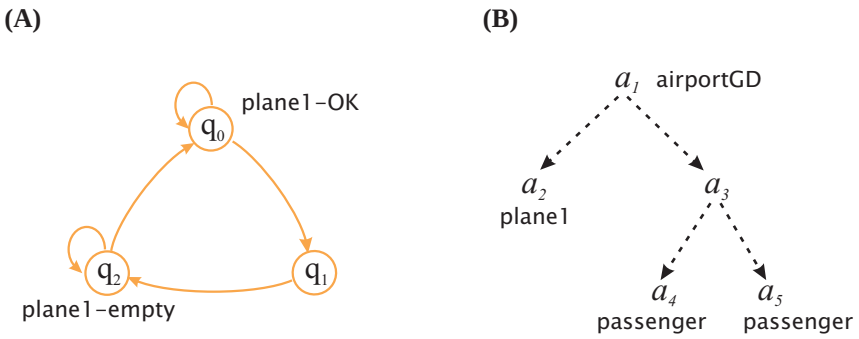

Figure 6: Modeling various aspects of multi-agent systems with possible worlds semantics: (A) temporal dimension, (B) structural dimension.

sitions defines which propositions are true in which states. An example of a temporal model is shown in Figure 6A. In state q_0, the plane number 1 is in order (plane1-OK), but something wrong may be detected even in the next moment (i.e. when the system executes a transition to state q_1, in which plane1-OK does not hold any more). If this becomes the case, all the passengers from plane 1 must be moved to another place (so that the plane is empty: plane1-empty) before the plane is repaired.

It is important to distinguish between the *computational structure*, defined explicitly in the model, and the *behavioral structure*, i.e. the model of how the system is supposed to behave in time [26]. Trees, obtained by unfolding from the computational structures, are the usual behavioral structures behind branching-time logics. In our case the transition system with hypermarkings as states, our computational structure, is always finite, whereas the tree of possible (infinite) paths (computations) that may occur in the system, is often infinite. Several operators are used to capture temporal properties in transition systems: $\bigcirc$ (*next*), $\Diamond$ (*sometime*), $\Box$ (*always*) and $\mathcal{U}$ (*until*). Thus, for example, $\Diamond$plane1-OK is a typical liveness formula that says that the first plane will be in order at some moment, while $\Box(\neg\text{plane1-OK} \rightarrow \Diamond\text{plane1-OK})$ expresses the fairness property that the plane is going to be consequently repaired after every breakdown. Furthermore, there are many alternative courses of action (paths) that can actually happen in the future. Typically, paths are interpreted as sequences of successive states of computations. There are basically two different traditions of how to tackle alternative paths. The linear-time temporal logic LTL treats each path as a separate alternative model of time [14]. The branching-time logic CTL [6, 8, 9] (Computation Tree Logic) collects all possible paths in a single model, and adds explicit *path quantifiers*: A (*for all paths*) and E (*there is a path*) to the language.

In this paper, we focus on CTL-like formalisms, with their explicit way of addressing tree-like semantic structures. CTL comes in two variants: in "vanilla" CTL, every occurrence of a temporal operator is preceded by exactly one path quantifier. In CTL*

no such restriction is imposed. CTL* is more expressive of the two (it strictly subsumes both CTL and LTL), but the "vanilla" version has slightly simpler semantics and some nice computational properties (e.g. model checking is linear in the size of the model and the length of the formula). $\mathsf{E}\Box\mathsf{plane1\text{-}OK}$ and $\mathsf{E}\bigcirc\mathsf{A}(\neg\mathsf{plane1\text{-}OK})\,\mathcal{U}\,\mathsf{plane1\text{-}empty}$ are example CTL properties that hold in state q_0 of the transition system presented in Figure 6A.

Of course, one may use LTL, CTL*, or a more expressive logic like ATL* [1, 2] or μ-calculus [17] instead. Essentially, the choice is a matter of taste, expressive power and complexity.

3.2 Structural Properties of Systems

Temporal logic allows for reasoning about possible evolution of abstract properties of systems that can be addressed through atomic propositions. In the context of mobile agents, agents locations and overall structure is also very interesting; if the system may evolve in time, we are interested in the structure of agents at each given moment. In the case of Petri hypernets, one can think here about properties that relate to the current subsumption structure of agents. For instance, proposition plane1-empty in the previous example was intended to capture that in a given state no traveller agent is *inside* the plane number 1.

However, rather than refer to such statements via atomic propositions, it is more practical and elegant to express the relevant structural properties by modal means. Modal logic offers much simpler way of reasoning about structures like the one presented in Figure 6B. Moreover, the agents' subsumption structure defined by a hypermarking in a PHN is always a *tree*, which suggests that a CTL-like logic can be a good formal tool to capture its properties. For instance, we may want to say that there is at least one passenger within the current agent ("there is a subsumption path with at least one passenger on it somewhere"). The "current" agent can in this case be understood as the agent on whose properties we currently focus.

To this end, we define new path quantifiers E_s (*there is a subsumption path*) and A_s (*for all subsumption paths*), plus spatial operators: $\textcircled{\downarrow}$ (*the agent one level down in the hierarchy*), $\boxed{\downarrow}$ (*all agents down, including the current agent*), $\downarrow$ (*all agents down until some property holds*), $\textcircled{\uparrow}$ (*the agent one level up*) $\boxed{\uparrow}$ (*all agents up, including the current agent*), and $\uparrow$ (*all agents up until some property holds*), analogous to the temporal operators of CTL. Now, the property of plane no. 1 being empty can be defined using the structural modalities: $\mathsf{plane1\text{-}empty} \equiv \boxed{\uparrow}\mathsf{A}_s\boxed{\downarrow}(\mathsf{plane1} \rightarrow \neg\mathsf{E}_s\textcircled{\downarrow}\top)$ which holds for every node of the hierarchy in Figure 7.

Note that in the logic of subsumption one should be able to reason not only about the "owned" agents, but also about the "owners". Adding past tense-style operators can, in general, pose complexity problems with respect to CTL model checking, cf. [26]. However, our subsumption hierarchies are always finite trees, so no additional computational cost is involved.

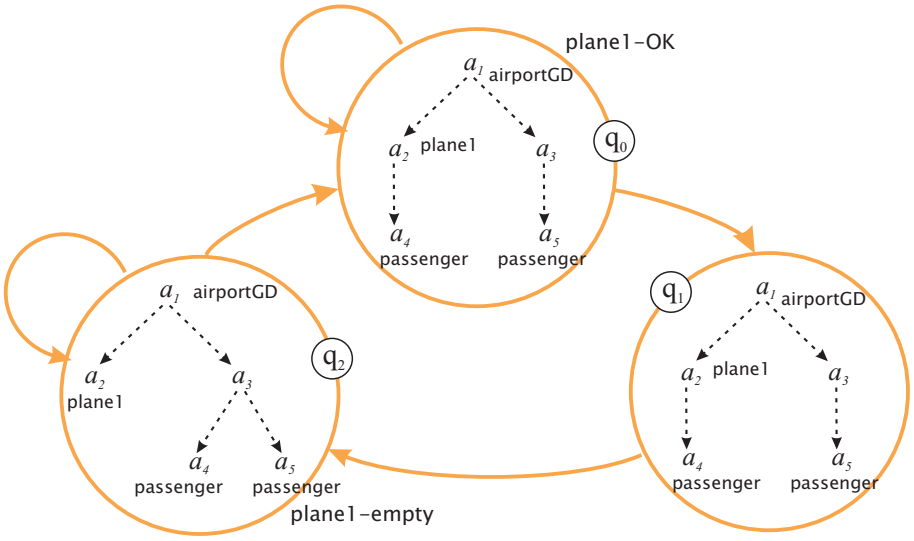

Figure 7: A hierarchy of agents evolving in time

Note also that, for every agent a, there is always exactly one maximal path upwards. Consequently, superposition path quantifiers are redundant.

3.3 Combining the Temporal and Structural Dimensions

In a Petri hypernet, every hypermarking defines a global state of the system, and induces subsumption structure among agents. As the hypermarking changes (because a transaction has been fired), the subsumption hierarchy may change as well. The picture is similar to the well-known BDI logics [23, 29]. There, temporal structures were embedded in epistemic positions of agents. Here, spatial agent structures are embedded in temporal positions of the system (i.e. states). The problem with such a presentation is that the models are *not* conventional Kripke structures: not only the possible worlds are complex ones, but also (some of) the accessibility relations are ternary rather than binary, because the two dimensions involved are not necessarily independent. Consider the model in Figure 7 that elaborates on the evolution of the airplane agents from Sections 3.1 and 3.2. The subsumption relation does depend on the state of the system: agent a_3 subsumes a_4 in state q_2, but not in state q_1! Thus, the resulting semantics over such a class of models cannot be a conventional Kripke semantics either.

It has been shown by Schild [25] that the semantics of BDI can be equivalently defined in terms of conventional Kripke structures with two binary modal relations, on which some structural restrictions are imposed. We follow that idea and introduce a class of models analogous to Schild's situation structures, that can be used to represent

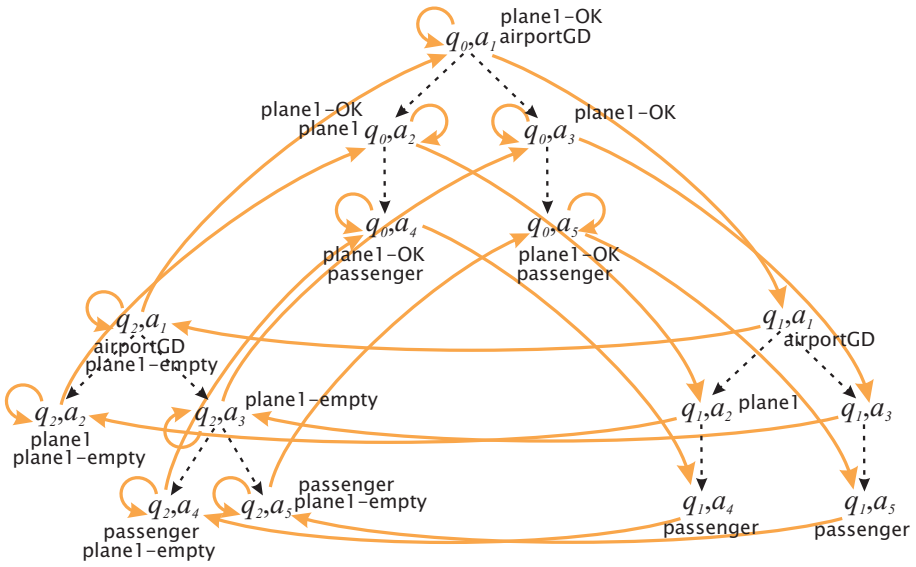

Figure 8: The "flattened" Kripke structure with two accessibility relations

the temporal and structural dimensions of agents in a PHN. The models, *agent-oriented hyper-transition systems*, are formally introduced in the next section. In Section 3.5, we show that a modal logic of time and space can be defined for such systems in a straightforward way.

3.4 Agent-Oriented Hyper-transition Systems

We define *agent-oriented hyper-transition systems* (AOHS in short) as 6-tuples: $M = \langle Q, q_0, \mathbb{A}\mathrm{gt}, \mathcal{R}, \mathcal{S}, \pi \rangle$, where:

- Q is a nonempty set of (temporal) *states* of the system, and $q_0 \in Q$ is a distinguished *initial state*.
- $\mathbb{A}\mathrm{gt}$ is a nonempty, finite set of *agents*.
- The set of possible worlds, or *agent-oriented states*, is defined as $\Omega = Q \times \mathbb{A}\mathrm{gt}$. An agent-oriented state $\langle q, a \rangle$ consists of q as its current temporal state, and a as the current agent. For an agent-oriented state $\omega = \langle q, a \rangle$, we define: $agent(\omega) = a$ and $state(\omega) = q$.
- $\mathcal{R}, \mathcal{S} \subseteq \Omega \times \Omega$ are two modal relations. $\mathcal{R}$ is the *temporal relation* that defines possible transitions between states, $\langle q, a \rangle \mathcal{R} \langle q', a' \rangle$ meaning that agent a at state q can evolve in a single step into agent a' at state q'. $\mathcal{S}$ is the *subsumption relation*

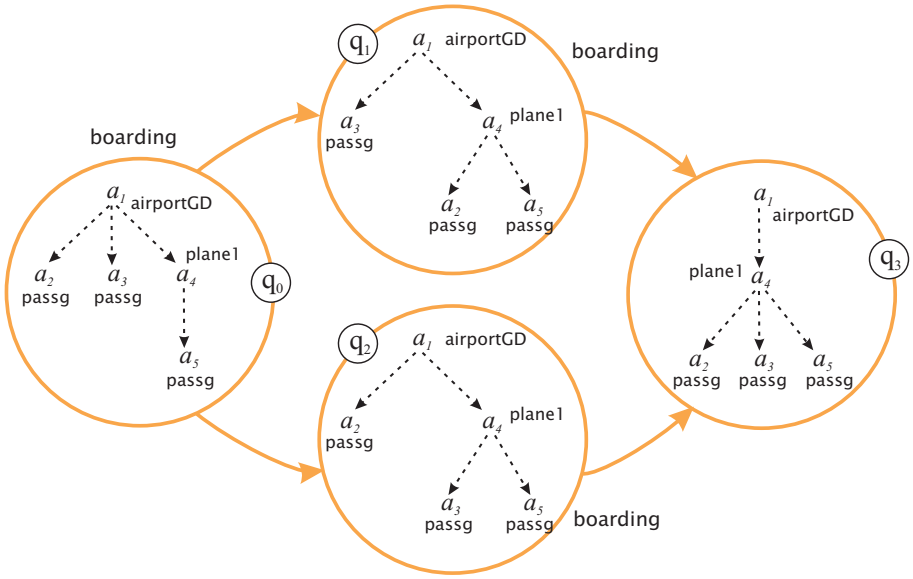

Figure 9: The hypernet agents from Section 2 evolving in time

that says which agent subsumes (or governs) whom at each particular moment. Thus, $\langle q, a\rangle \mathcal{S} \langle q', a'\rangle$ means that agent a at state q subsumes (or controls) agent a' at state q'. We require relations $\mathcal{R}, \mathcal{S}$ to satisfy the following structural conditions:

1. $\langle q, a\rangle \mathcal{S} \langle q', a'\rangle \Rightarrow q = q'$: agents subsume each other *within states*,
2. $\langle q, a\rangle \mathcal{R} \langle q', a'\rangle \Rightarrow a = a'$: $\mathcal{R}$ models evolution of each agent *separately*,
3. $\langle q, a_1\rangle \mathcal{R} \langle q', a_1\rangle \Rightarrow \langle q, a_2\rangle \mathcal{R} \langle q', a_2\rangle$: the temporal evolution is the same for all agents (time is global).

- Finally, $\pi : \Pi \to \Omega$ is a valuation of atomic propositions from a given set Π. Thus, for every proposition $p \in \Pi$, $\pi(p)$ defines the set of agent-oriented states in which p holds.

Remark 3 *Note that the above properties of $\mathcal{R}, \mathcal{S}$ imply that, formally speaking, we may as well use a single modal accessibility relation $\mathcal{R}'$ in our models, and define temporal and subsumption accessibility on top of it: $\mathcal{R} = \{\langle \omega, \omega'\rangle \mid agent(\omega) = agent(\omega')\}$ and $\mathcal{S} = \{\langle \omega, \omega'\rangle \mid state(\omega) = state(\omega')\}$. We believe that two separate relations make the models conceptually clearer and easier to read, and therefore we will use this kind of presentation here.*

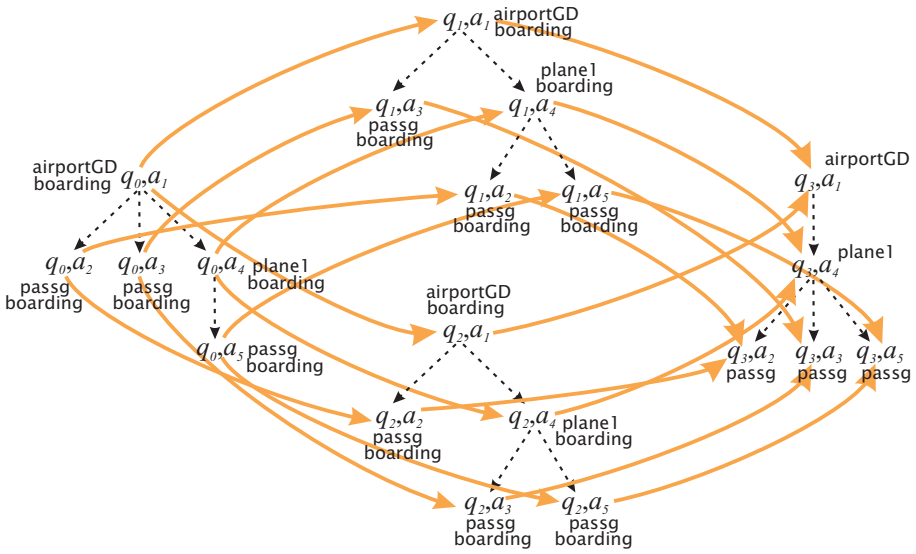

Figure 10: Kripke structure for the hypernet agents

The notion of a *reachable* world is defined as usual. Note that, due to (3), for every reachable agent-oriented state $\langle q, a\rangle$, we have $\langle q, a'\rangle$ also being reachable for any agent a'. This captures the "no creation, no destruction of agents" principle.

We need hyper-transition systems to represent temporal evolution of structured systems of agents defined via Petri hypernets. Thus, due to the properties of Petri hypernets, we will additionally require that:

4. $\mathcal{S}$ forms a tree at every temporal state. That is, $G_q = \langle \Omega_q, \mathcal{S} \cap (\Omega_q \times \Omega_q)\rangle$, where $\Omega_q = \{\omega \mid state(\omega) = q\}$, is a tree for every $q \in Q$.

Figure 8 presents an AOHS that models the evolving hierarchy of agents from Figure 7. In Figure 9, we present how the hierarchy of the "hypernet" agents from Figure 3 (Section 2) can evolve. The corresponding AOHS is presented in Figure 10. We will refer to the transition system from Figure 10 as M_1 in further examples.

A *ρ-path* in M is a sequence of agent-oriented states that follow relation ρ, that is, a sequence $\omega_0\omega_1\omega_2\ldots$ such that $\omega_i \rho \omega_{i+1}$ for every $i = 0, 1, 2, \ldots$. In what we consider, ρ can be one of the three relations: $\mathcal{R}, \mathcal{S}, \mathcal{S}^{-1}$ (thus, we can have paths that refer to an agent's evolution in time, agent subsumption chains, and agent superposition chains). A *full ρ-path*, or *run*, is a path that is either infinite or ends with a state ω that is blocked: $\omega\rho\omega'$ for no ω'. Finally, we denote the ith state in path Λ by $\Lambda[i]$ (starting from $i = 0$).

3.5 The Logic of CTL^2

We now formally introduce a logic, in which both the temporal and the structural dimensions are captured by CTL-style modalities. The logic of CTL^2 (with respect to a set of atomic propositions Π) is defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}\bigcirc\varphi \mid \mathsf{E}\Box\varphi \mid \mathsf{E}\varphi\mathcal{U}\varphi \mid \mathsf{A}\bigcirc\varphi \mid \mathsf{A}\Box\varphi \mid \mathsf{A}\varphi\mathcal{U}\varphi \mid \mathsf{E}_s\textcircled{\downarrow}\varphi \mid \mathsf{E}_s\boxed{\downarrow}\varphi \mid \mathsf{E}_s\varphi\downarrow\varphi \mid \mathsf{A}_s\textcircled{\downarrow}\varphi \mid \mathsf{A}_s\boxed{\downarrow}\varphi \mid \mathsf{A}_s\varphi\downarrow\varphi \mid \textcircled{\uparrow}\varphi \mid \boxed{\uparrow}\varphi \mid \varphi\uparrow\varphi,$$

where $p \in \Pi$. The semantics of CTL^2 is defined as follows:

$M, \omega \models p$	iff $\omega \in \pi(p)$, for an atomic proposition p;
$M, \omega \models \neg\varphi$	iff $M, \omega \not\models \varphi$;
$M, \omega \models \varphi \wedge \psi$	iff $M, \omega \models \varphi$ and $M, \omega \models \psi$;
$M, \omega \models \mathsf{E}\bigcirc\varphi$ $M, \omega \models \mathsf{E}_s\textcircled{\downarrow}\varphi$ $M, \omega \models \textcircled{\uparrow}\varphi$	iff there is a full $\mathcal{R}$-path Λ ($\mathcal{S}$-path, $\mathcal{S}^{-1}$-path, respectively) with the beginning state $\Lambda[0] = \omega$, such that $M, \Lambda[1] \models \varphi$;
$M, \omega \models \mathsf{E}\Box\varphi$ $M, \omega \models \mathsf{E}_s\boxed{\downarrow}\varphi$ $M, \omega \models \boxed{\uparrow}\varphi$	iff there is a full $\mathcal{R}$-path Λ ($\mathcal{S}$-path, $\mathcal{S}^{-1}$-path, respectively) with the beginning state $\Lambda[0] = \omega$, such that $M, \Lambda[i] \models \varphi$ for all $i \geq 0$;
$M, \omega \models \mathsf{E}\varphi\mathcal{U}\psi$ $M, \omega \models \mathsf{E}_s\varphi\downarrow\psi$ $M, \omega \models \varphi\uparrow\psi$	iff there is a $\mathcal{R}$-path Λ ($\mathcal{S}$-path, $\mathcal{S}^{-1}$-path, resp.) with $\Lambda[0] = \omega$, such that there is $i \geq 0$ with $M, \Lambda[i] \models \psi$ and for all j such that $0 \leq j \leq i$ we have $M, \Lambda[j] \models \varphi$.

The semantics of universal path quantifiers A and A_s is defined analogously (i.e., "there is a path" is replaced by "for all paths"). Additionally, the "sometime" operator is defined as: $\Diamond\varphi \equiv \top\mathcal{U}\varphi$. Formula φ is *valid* in model M iff $M, \langle q_0, a\rangle \models \varphi$ for every $a \in \mathbb{A}\mathrm{gt}$.

Example 1 *Consider system M_1 from Figure 10. Now, formula $\neg\mathsf{E}_s\textcircled{\downarrow}\top$ expresses the property that the "current" agent is empty, i.e. it contains no further agents. This property always holds for every passenger agent: that is, $\mathsf{passg} \rightarrow \mathsf{A}\Box\neg\mathsf{E}_s\textcircled{\downarrow}\top$ is valid in M_1. Other formulae, valid in M_1, are: $\mathsf{A}\Box(\mathsf{boarding} \wedge \mathsf{airportGD} \rightarrow \mathsf{E}_s\textcircled{\downarrow}\mathsf{passg})$: while boarding, there are passengers at the Gdansk airport, and $\mathsf{A}\Box\,\boxed{\uparrow}\mathsf{A}_s\boxed{\downarrow}(\mathsf{passg} \rightarrow \mathsf{A}\Diamond\textcircled{\uparrow}\mathsf{plane1})$: all passengers are going to sit in plane 1 eventually.*

Example 2 *Suppose that the Gdańsk airport and the Milan airport are labeled with agent propositions $\mathsf{airportGD}$ and $\mathsf{airportMIL}$, respectively. Let plane be a proposition that labels all the airplane agents. Now, $\mathsf{passg} \rightarrow \mathsf{A}\Box(\textcircled{\uparrow}\mathsf{airportGD} \rightarrow \mathsf{E}\Diamond\textcircled{\uparrow}(\mathsf{plane} \wedge \mathsf{A}\Diamond\textcircled{\uparrow}\mathsf{airportMIL}))$ says that a passenger, whenever in Gdansk, can possibly board a connection that is bound for Milan.*

3.6 CTL^2 as a Combination of Logics

CTL^2 defined by the above monolithic definition can also be explained as a certain *combination* of the two components describing the *temporal* and the *spatial* dimensions of Petri hypernets, respectively. Many forms of (modal) logic combinations have been proposed in the literature. The simplest method introduced to combine modal logics was *fusion* (cf. [18]). Semantically speaking, it combines two Kripke semantics by putting their accessibility relations side by side. Fusion does not introduce any interference between the logics being combined and as such is not appropriate for our purposes. Another, very popular method called *product* or sometimes *join* (cf. [15]) allows for certain level of semantical interaction by "interleaving" the accessibility relations. Still, it is not sufficiently flexible to model CTL^2.

The most famous logic combination method called *fibring* (cf. [13]) was developed by Dov Gabbay. It allows us to associate to each possible world within a model of one logic a model of another logic via so called *fibring function*. While evaluating a *combined language* formula the fibring function allows us to go "back and forth" between semantics of the logics being combined.

As it turns out, to adequately capture the semantics of CTL^2, it is best to apply a method which lays somewhere between the product and the fibring constructions. Below we show how to obtain CTL^2 as a combination of its temporal and spatial CTL-like components. We denote these components by CTL^2_T and CTL^2_S respectively. We concentrate on the semantic part of the construction, since at the syntactic level the "interleaved formulae" of CTL^2 are obtained by simply taking the union of the formation rules for CTL^2_T and CTL^2_S (as in the case of both product and fibring).

For any CTL^2_T model $M = \langle Q, q_0, \mathcal{R}, \pi\rangle$ let us consider an arbitrary family (fibring function) $\mathcal{F}_M = \langle\langle A_M, \mathcal{S}^q, \pi^q\rangle \mid q \in Q\rangle$ of CTL^2_S models. Please note, that we assume that all the models in $\mathcal{F}_M$ share the same set of states. Using the model M and the fibring function $\mathcal{F}_M$ we can define a *combined model* as a tuple $\langle Q\times A, q_0, \overline{\mathcal{R}}, \overline{\mathcal{S}}, \overline{\pi}\rangle$, where

- $\overline{\mathcal{R}} = \{\langle\langle q_1, a\rangle, \langle q_2, a\rangle\rangle \mid q_1 \mathcal{R} q_2\}$
- $\overline{\mathcal{S}} = \{\langle\langle q, a_1\rangle, \langle q, a_2\rangle\rangle \mid a_1 \mathcal{S}^q a_2\}$
- $\overline{\pi}(p) = \begin{cases} \pi(p) \times A_M & \text{for } p \in \Pi_T \\ \{\langle q, a\rangle \mid a \in \pi^q(p) \wedge q \in Q\} & \text{for } p \in \Pi_S \end{cases}$

where Π_T and Π_S denote the appropriate sets of temporal and spatial/structural atomic propositions. The issue of a particular choice of atomic propositions was already discussed in Remark 2.

We take the class of all *combined models* as the class of models of the resulting logic. From the construction it is clear that combined models are equivalent to the *agent-oriented hyper-transition systems* as defined in Section 3.4. The semantic interpretation of formulae of CTL^2 is obtained as the union of semantic rules for CTL^2_T and CTL^2_S.

4 Model Checking Properties of Mobile Agents

The *model checking* problem asks whether a given formula φ holds in a given model M and state q. It is often convenient to require the model checking algorithm to return the set of states in M that satisfy φ. Any CTL model checking algorithm (e.g. [7]) can be easily adapted to handle CTL^2 formulae: spatial properties are processed in the same way as their temporal counterparts, only $\mathcal{S}$ or $\mathcal{S}^{-1}$ is taken instead of $\mathcal{R}$ as the reachability relation, when appropriate . Thus, the results for CTL model checking (cf. [26]) apply to CTL^2 as well.

Proposition 4 *Let l be the length of a CTL^2 formula φ, and $\overline{m}$ be the cardinality of the "densest" modal relation in an agent-oriented hyper-transition system M (i.e. the maximal number of pairs in $\mathcal{R}, \mathcal{S}$). Model checking φ in M is* PTIME*-complete and can be done in time $O(\overline{m}l)$. Moreover, the structure complexity (i.e. the complexity when the formula is assumed to be fixed and is not a parameter of the problem any more) of CTL^2 model checking is* NLOGSPACE.

We will see in Section 4.1 that the result actually promises less than it suggests, because the AOHS derived from Petri hypernets are usually large. However, this problem is well known for the original CTL too: CTL models are usually exponentially large in terms of a higher-level description of the problem domain—yet model checking turns out to be feasible in many cases [20, 21].

4.1 From Hypernets to Hyper-transition Systems

Let $H = \langle \mathcal{N}, m \rangle$ be a hypernet. CTL^2 properties can be defined in terms of the AOHS that includes all reachable markings of H, and transitions that can occur in H. Formally, $aohs(H) = \langle Q_H, q_0, \mathbb{A}\text{gt}_H, \mathcal{R}_H, \mathcal{S}_H, \pi_H \rangle$, is called an agent-oriented hypertransition system associated with H, and defined as follows.

- $\mathbb{A}\text{gt}_H = \mathcal{N}$ (agents are identified with nets of H);
- Q_H and $\mathcal{R}_H$ are defined recursively: (1) $m \in Q_H$, (2) if a transaction can be fired in hypernet $\langle \mathcal{N}, m_1 \rangle$, $m_1 \in Q_H$, yielding $\langle \mathcal{N}, m_2 \rangle$, then: (i) $m_2 \in Q_H$ and (ii) $\langle m_1, a \rangle \mathcal{R}_H \langle m_2, a \rangle$ for all $a \in \mathbb{A}\text{gt}_H$;
- $\langle m, a_1 \rangle \mathcal{S}_H \langle m, a_2 \rangle$ iff $m(a_1) \in P_{a_2}$, where P_a is the set of places of agent a;
- $q_0 = m$: the initial state is defined by the current hypermarking m;
- if $p \in \Pi_{ag}$ then $\pi(p) = Q_H \times \lambda_{ag}(p)$: agent labels in H yield agent-identifying propositions in $aohs(H)$;
- If $p \in \Pi_{pl}$ then $\pi(p) = \{m \mid \forall \mathfrak{p} \in \lambda_{pl}(p)\ \exists N \in \mathcal{N}\ .\ m(N) \in \mathfrak{p}\} \times \mathbb{A}\text{gt}_H$. That is, p distinguishes those hypermarkings, according to which every place from $\lambda_{pl}(p)$ is inhabited by some token.

Note that the place propositions from Π_{pl} are purely *temporal* properties which only depend on the current temporal state. On the other hand, the agent propositions (elements of Π_{ag}) are purely *structural* properties depending on the current agent—the spatial/structural state.

Proposition 5 *$aohs(H)$ is an* AOHS *for every* PHN *H.*

Proof. First, the way relations $\mathcal{R}, \mathcal{S}$ are constructed in $aohs(H)$ implies structural conditions (1)–(3). Moreover, it was shown in [3] that the tree structure of agents is preserved when a transaction is fired. Therefore condition (4) holds as well. ■

Remark 6 *The reverse does not hold: there are* AOHS *that do not correspond to any* PHN*. For example, it is not possible to change the root agent in a hypernet (i.e. the agent that contains all other agents, directly or indirectly), while such an* AOHS *is viable according to the definition.*

The following result shows that AOHS are large in comparison with their PHN counterparts. In fact, the increase of size when we shift from a hypernet H to the transition system $aohs(H)$ is (slightly) more than exponential, which follows directly from Remark 1.

Proposition 7 *For a hypernet H with n places in total, $aohs(H)$ includes at most $O(n)2^{O(n \log n)}$ agent-oriented states, and the bound is tight.*

Note that the cardinality of modal relations in AOHS M is $O(n_M^2) = O(k^2 n^{2k})$. Thus, a CTL-based model checking algorithm will have a time complexity of $O(k^2 n^{2k} l)$. It seems worth pointing out that the size of the $aohs(H)$ becomes polynomial when the number of agents is fixed or bounded. Unfortunately, decomposing complex systems into *many* individual agents is an obvious methodology and in fact it is one of the main advantages that Petri hypernets can offer.

4.2 Model Checking Petri Hypernets

Having defined the correspondence between AOHS and (labeled) PHN, we can say what it means that a CTL^2 formula holds for net a in a hypernet $H = \langle \mathcal{N}, m \rangle$.

Definition 1 $H, a \models \varphi \quad \text{iff} \quad aohs(H), \langle m, a \rangle \models \varphi$.

Proposition 8 *Model checking* CTL^2 *formulae over* PHN *is* PSPACE*-hard.*

Proof. We show this through a reduction of the QSAT problem (satisfiability for quantified Boolean formulae [22]). In QSAT, we are given k propositional variables $p_1, ..., p_k$ partitioned into sets $P_1, P_2, ...$, and a formula $\Phi \equiv \exists P_1 \forall P_2 \exists P_3 ... \varphi$, where φ is a Boolean combination of $p_1, ..., p_k$. We construct a hypernet H that includes

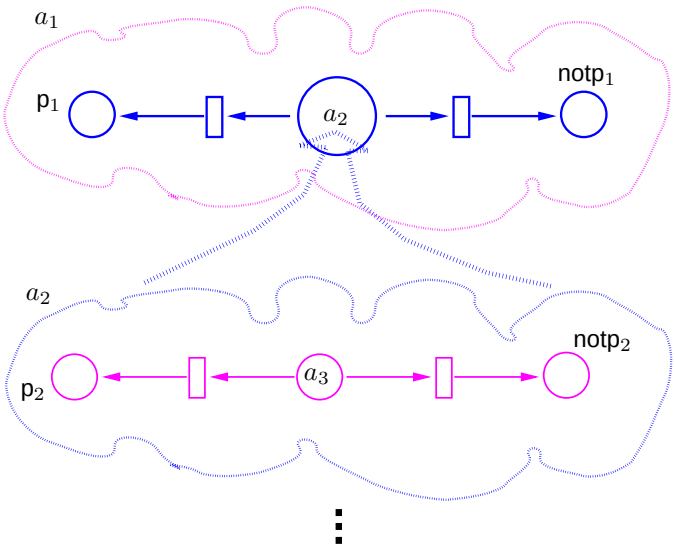

Figure 11: A hypernet generating an arbitrary assignment for $p_1, ..., p_k$

$k+1$ agents, each next agent placed inside the previous one. The task of agent a_i for $i = 1, ..., k$ is to "declare" variable p_i true or false; the agent does it by moving "his" sub-agent to a place where proposition p_i (or proposition notp_i) holds. Initially, all the agents are "in the middle" of their super-agents, i.e. no decision has been taken. Agent a_{k+1} is just an empty net, serving as a token to endow a_k with an internal state. The structure of the net is depicted in Figure 11. We define auxiliary formulae $\mathsf{ready}_i \equiv \bigwedge_{j=1}^{i}(\mathsf{p}_j \vee \mathsf{notp}_j) \wedge \bigwedge_{j=i+1}^{k} \neg(\mathsf{p}_j \vee \mathsf{notp}_j)$, saying that agents $a_1, ..., a_i$ have done their job, and the rest have not.

We also assume that φ has been transformed so that negations apply to atomic propositions only. We propose the following translation of Φ:

$$tr(\exists p_i \Psi) = \mathsf{E}\bigcirc(\mathsf{ready}_i \wedge tr(\Psi)) \qquad tr(\forall p_i \Psi) = \mathsf{A}\bigcirc(\mathsf{ready}_i \rightarrow tr(\Psi))$$
$$tr(\varphi \wedge \psi) = tr(\varphi) \wedge tr(\psi) \qquad tr(\varphi \vee \psi) = tr(\varphi) \vee tr(\psi)$$
$$tr(p_i) = \mathsf{p}_i \qquad tr(\neg p_i) = \mathsf{notp}_i$$

Note that, for the nets in H, $tr(\exists p_i \Psi)$ holds if and only if a_i is the only agent to make the next move, and it *can* make a move after which Ψ holds. Similarly, $tr(\forall p_i \Psi)$ holds if and only if Ψ holds after every choice of a_i. Moreover, φ is transformed into a CTL^2 formula that requires values of propositions $p_1, ..., p_k$ to have been declared (on the right levels) in such a way that they make φ true. Thus, Φ holds iff $H, a_1 \models tr(\Phi)$. Moreover, H includes only $O(k)$ places, and the length of $tr(\Phi)$ is $O(k^2 + |\varphi|)$, which concludes the proof. ■

For the upper bound, let n_K, m_K be the numbers of nodes and transitions in a Kripke structure (e.g. an AOHS). It has been proved in [19] (Theorem 5.7) that formulae of CTL can be model-checked in space $O(l \log^2((n_K + m_K)l))$. By Proposition 7, CTL^2 can be model-checked in space $O(l^3 n_H^4)$, where n_H is the number of places in the underlying hypernet H. As [19] presents an analogous result concerning CTL* model checking, and LTL can be seen as a fragment of CTL*, we can state the following.

Theorem 9 *Model checking* CTL^2 *over Petri hypernets is* PSPACE*-complete. The same complexity bounds apply if we use* LTL *or* CTL* *to reason about the temporal and/or structural dimension of Petri hypernets.*

The PSPACE-completeness is actually what one should expect from a problem of this kind [10, 11]. We note that the problem is no worse than for the logic of mobile ambients [5], and for the standard temporal logics CTL, LTL and CTL* (when the input is given as a combination of smaller components [26]). This hints that automatic verification of agent properties via Petri hypernets and modal logics like CTL^2 should be feasible after all.

5 Conclusions

We have shown that one indeed, as advocated in [12], can use a combination of two modal logics, one to cope with the temporal, the other with the structural aspects, in order to specify the properties of mobile agents. The logic we have obtained cannot be seen as one of the simple combinations considered in [12], though. Instead, we blend some aspects of *product* and *fibring*. We treat our choice of logics to address the temporal and the spatial dimension as somewhat arbitrary. In a way, one can treat the actual logics as parameters of the resulting framework. In this paper, we used CTL to address both dimensions, but most other temporal logics can be used instead in a natural way. It can be interesting to study what properties can be specified using other, more expressive logics – most notably μ-calculus [17], ATL [1, 2], and logics of strategic ability under incomplete information [16, 27].

In CTL^2, atomic propositions cope purely with one of the aspects, either temporal or spatial. In the case of hypernets it seems very natural to consider atomic propositions of the form: "the agent is at gate G7", which refer to both the current temporal state and the current agent. Thus, finer-grained models and a more general combination of logics seem to be desirable – which suggests another promising direction for future research. This shows also that Petri hypernets offer a rich modeling framework, one which has not been matched yet by a sufficiently expressive logical counterpart.

References

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109. IEEE Computer Society Press, 1997.

[2] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time Temporal Logic. *Journal of the ACM*, 49:672–713, 2002.

[3] M. Bednarczyk, L. Bernardinello, W. Pawłowski, and L. Pomello. Modelling mobility with Petri hypernets. In *Proceedings of WADT 2004*, volume 3423 of *Lecture Notes in Computer Science*, pages 28–44, 2004.

[4] L. Cardelli and A. D. Gordon. Mobile ambients. *Electr. Notes Theor. Comput. Sci.*, 10, 1997.

[5] W. Charatonik, S. Dal-Zilio, A. D. Gordon, S. Mukhopadhyay, and J.-M. Talbot. Model checking mobile ambients. *Theor. Comput. Sci.*, 308(1-3):277–331, 2003.

[6] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of Logics of Programs Workshop*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71, 1981.

[7] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[8] E. Emerson and J. Halpern. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[9] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 995–1072. Elsevier Science Publishers, 1990.

[10] J. Esparza. Decidability of model checking for infinite-state concurrent systems. *Acta Inf.*, 34(2):85–107, 1997.

[11] J. Esparza. Decidability and complexity of Petri net problems. An introduction. In G. Rozenberg and W. Reisig, editors, *Lectures on Petri Nets I: Basic Models. Adv. in Petri Nets*, volume 1491 of *LNCS*, pages 374–428. Springer Verlag, 1998.

[12] M. Franceschet, A. Montanari, and M. de Rijke. Model checking for combined logics with an application to mobile systems. *Autom. Softw. Eng.*, 11(3):289–321, 2004.

[13] D. M. Gabbay. *Fibring Logics*, volume 38 of *Oxford Logic Guides*. Oxford Univesity Press, 1998.

[14] D. M. Gabbay, A. Pnuelli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of POPL'80*, pages 163–173, 1980.

[15] D. M. Gabbay and V. Shehtman. Products of modal logics, part 1. *Logic Journal of the IGPL*, 6(1):73–146, 1998.

[16] W. Jamroga and W. van der Hoek. Agents that know how to play. *Fundamenta Informaticae*, 63(2–3):185–219, 2004.

[17] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[18] M. Kracht and F. Wolter. Properties of independently axiomatizable bimodal logics. *Journal Of Symbolic Logic*, 54 (4):1469–1485, 1991.

[19] O. Kupferman, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[20] K. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. Kluwer Academic Publishers, 1993.

[21] K. McMillan. Applying SAT methods in unbounded symbolic model checking. In *Proceedings of CAV'02*, volume 2404 of *Lecture Notes in Computer Science*, pages 250–264, 2002.

[22] C. Papadimitriou. *Computational Complexity*. Addison Wesley : Reading, 1994.

[23] A. Rao and M. Georgeff. Modeling rational agents within a BDI-architecture. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484, 1991.

[24] W. Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[25] K. Schild. On the relationship between BDI logics and standard logics of concurrency. *Autonomous Agents and Multi Agent Systems*, pages 259–283, 2000.

[26] P. Schnoebelen. The complexity of temporal model checking. In *Advances in Modal Logics, Proceedings of AiML 2002*. World Scientific, 2003.

[27] P. Y. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2), 2004.

[28] R. Valk. Petri nets as token objects: An introduction to elementary object nets. In W. van der Aalst and E. Best, editors, *Applications and Theory of Petri Nets 1998, Proceedings*, volume 1420 of *LNCS*, pages 1–25. Springer-Verlag, 1998.

[29] M. Wooldridge. *Reasoning about Rational Agents*. MIT Press : Cambridge, Mass, 2000.

Impressum

Publisher: Institut für Informatik, Technische Universität Clausthal
Julius-Albert Str. 4, 38678 Clausthal-Zellerfeld, Germany
Editor of the series: Jürgen Dix
Technical editor: Wojciech Jamroga
Contact: wjamroga@in.tu-clausthal.de
URL: http://www.in.tu-clausthal.de/~wjamroga/techreports/
ISSN: 1860-8477

The IfI Review Board