

Model Checking Dynamic Allocation and Deallocation*

DINO DISTEFANO, AREND RENSINK, JOOST-PIETER KATOEN

Department of Computer Science, University of Twente

P.O. Box 217, 7500 AE Enschede, The Netherlands

E-mail: {ddino, rensink, katoen}@cs.utwente.nl

Fax: +31-53-4893247

Abstract

This paper proposes Allocational Temporal Logic (ATL) as a formalism to express properties concerning the dynamic allocation (birth) and de-allocation (death) of entities, such as the objects in an object-based system. The logic is interpreted on History-Dependent Automata, extended with a symbolic representation for certain cases of unbounded allocation. The paper also presents a simple imperative language with primitive statements for (de)allocation, with an operational semantics, to demonstrate the kind of behaviour that can be modelled. The main contribution of the paper is a tableau-based model checking algorithm for ATL, along the lines of Lichtenstein and Pnueli's algorithm for LTL.

*CTIT Technical Report TR-01-40. Department of Computer Science, University of Twente.

Contents

1	Introduction	2
2	Allocational temporal logic	4
2.1	Syntax	4
2.2	Semantics	6
2.3	Folded allocational sequences	6
2.4	Relating unfolded and folded allocational sequences	7
3	Allocational Büchi automata	9
3.1	ABA	9
3.2	HABA	10
3.3	The duality between ABA and HABA	13
4	Programming allocation and deallocation	16
4.1	Syntax	16
4.2	Concrete semantics	18
4.3	Symbolic semantics	21
4.4	Relating the concrete and symbolic semantics	23
5	Model-checking ATL	24
5.1	Duplication	25
5.2	Valuations	26
5.3	Tableau graph	28
5.4	Complexity	39
6	Related work	42
7	Conclusions and future work	43
A	Proofs of Section 2	46
B	Proofs of Section 3	46
C	Proofs of Section 4	49
D	Proofs of Section 5	52

1 Introduction

One of the aspects of computation that state-of-the-art model checking does not deal with very well is that of dynamic *allocation* and *deallocation* (birth and death) of entities. This is especially true if the number of entities is not known beforehand, or even unbounded. Nevertheless, allocation and deallocation are fundamental concepts in many fields of computer science. For example, in object-orientation, systems are composed by dynamic objects (*entities*) that can be created (*allocated*) in an arbitrary number during the computation. Moreover objects can be destroyed (*deallocated*) e.g., by a garbage collector. Similarly, instances of

method calls (used by objects for interaction) are *entities* that are *allocated* (created) at the moment of the invocation and that are *deallocated* (destroyed) when the body of the method is completed. Another significant area where a concept like freshness is certainly relevant is security. In this field, properties of systems related with *fresh* secure resources as keys are mostly investigated. Apart from these two specific disciplines, it is not difficult to observe that almost every computer system is dealing at run time with activities like allocation and deallocation of resources (memory, channels, processes, etc.).

Though there are now calculi (such as the π -calculus [19]) that can express the generation of fresh names, as well as models (such as history-dependent automata [20]) that can describe both the birth and the death of entities, what has been missing so far is a logic where these concepts are captured as primitives; a logic that should be as fundamental to reasoning about dynamic allocation as standard propositional logic is to reasoning about a fixed state space. A formalism that, in a natural way, would allow the specification of properties like

- *a fresh entity will always eventually be allocated or*
- *before a particular entity is deallocated, two new entities will be allocated.*

Returning to the example of object-oriented systems, we would like to express properties like *every object in the current state will be eventually deallocated* or *the number of running objects will never be less than two*. For security: *an authentication server never provides already used secret session keys*. In fact, in protocols involving shared-key cryptography, session keys must be fresh for every communication session between principals (agents). In general, authentication servers are trusted to generate new keys in a proper manner, however, it is easy to imagine a possible naive or, even faulty, implementation of such servers.

An attempt to formulate such a logic is presented in this paper. Called *allocational temporal logic* (ATL), it has the following features:

- Entity variables x, y , interpreted by a mapping to the entities existing (i.e., *alive*) in a given state. The interpretation is *partial*: a variable not mapped onto an existing entity stands for an entity that has *died*.
- Entity equations $x = y$ (where x, y are entity variables), asserting that x and y refer to the same entity. This cannot hold if either x or y has died; hence the entity equations express a *partial equivalence* of entity variables (symmetric and transitive, but not reflexive).
- Entity quantification $\forall x.\phi$, which holds in a given state if ϕ holds always, regardless of the interpretation of x , provided that x is alive.
- A predicate x *new* to express that the entity referred to by x is *fresh*, i.e., newly born.

In addition, ATL has the standard LTL temporal operators (although a branching-time version with CTL temporal operators can be defined in a rather similar way).

The logic is interpreted over so-called *high-level allocational Büchi automata* (HABA), which extend *history-dependent automata* [20] with a predicate for the *unboundedness* of (the number of entities in) a state, and with a (generalized) Büchi acceptance condition. As for history-dependent automata, a crucial point is that entity identity is *local* to a state. Correspondence between the identity of the same entity in two different states is ensured by a mechanism of re-mapping.

We use HABAs to model the behaviour of systems. As an example, we define a small language whose main features are the allocation and deallocation of entities. Although the number of entities allocated by a program in this language can be unbounded, the operational semantics yields a finite HABA. This is a significant condition for the application of model checking.

Together with the logic ATL, the main contribution of this paper is that the model-checking problem for ATL is shown to be decidable on HABA. In particular, we present a tableau-based model-checking algorithm that decides whether a given ATL-formula holds for a given HABA. Our algorithm extends the tableau-based algorithm for LTL [17]. To the best of our knowledge, this yields the first approach to effectively model-check models with an unbounded number of entities. This is of particular interest to e.g. the verification of object-oriented systems in which the number of objects is typically not known in advance and may be even unbounded. Hopefully, in this context, the algorithm can be adapted to more domain specific temporal logics like BOTL [9].

Organisation of the paper. This paper introduces the logic (Section 2) and its automata (Section 3), as well as a simple imperative language, featuring statements for the allocation and deallocation of entities, with an operational semantics in terms of HABA (Section 4). The latter provides an intuition about the setup and the sort of behaviour that HABA can model. Apart from the presentation of the logic ATL, the main contribution of the paper is the proof of its model checking property (Section 5). A discussion about related work follows (Section 6), while future work and conclusion completes the paper (Section 7). Proofs are in the appendix.

2 Allocational temporal logic

2.1 Syntax

In the following we assume the existence of

- a countable universe of logical variables $LVar$, ranged over by x, y, z ;
- a countable universe of entities Ent , ranged over by e, e', e_1, \dots .

Allocational Temporal Logic (ATL) is an extension of propositional LTL [22] that allows existential quantification over logical variables that can denote entities, or may be undefined. For $x \in LVar$, the syntax of ATL is defined by the following grammar:

$$\phi ::= x \text{ new} \mid x = x \mid \exists x. \phi \mid \neg \phi \mid \phi \vee \phi \mid \mathbf{X}\phi \mid \phi \mathbf{U} \phi$$

The operators have the following intuitive meaning. Formula $x \text{ new}$ holds if the entity denoted by x is fresh in the current state, i.e., the entity denoted by x did not exist in the previous state. Formula $x = y$ holds if variables x and y denote the same entity in the current state; $x = x$ is violated if x is undefined, i.e., if x does not denote any current entity. $\exists x. \phi$ is valid in the current state if there exists an entity for which ϕ holds if assigned to x . Negation and disjunction are the standard operators of classical propositional logic, while \mathbf{X} (next) and \mathbf{U} (until) are the standard LTL operators.

Notation In the following we will use some useful abbreviations:

- | | | | | | |
|---------------------------|-----|--------------------------|-------------------------------|-----|--|
| • $x \neq y$ | for | $\neg(x = y)$ | • x alive | for | $x = x$ |
| • x dead | for | $x \neq x$ | • x old | for | x alive $\wedge \neg(x$ new) |
| • $\forall x.\phi$ | for | $\neg\exists x.\neg\phi$ | • tt | for | $\forall x.(x$ alive) |
| • ff | for | \neg tt | • $\phi \wedge \psi$ | for | $\neg(\neg\phi \vee \neg\psi)$ |
| • $\phi \Rightarrow \psi$ | for | $\neg\phi \vee \psi$ | • $\phi \Leftrightarrow \psi$ | for | $(\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$ |
| • $F\phi$ | for | tt $\cup \phi$ | • $G\phi$ | for | $\neg F\neg\phi$ |

Example 2.1. Properties concerning dynamic allocation and deallocation can be formalised in ATL. For example:

- New entities are always available:

$$G(\exists x.x \text{ new})$$

- The number of entities that are alive never exceeds 2:

$$G(\forall x.\forall y.\forall z.(x = y \vee x = z \vee y = z))$$

- The number of alive entities will never be less than 2:

$$G(\exists x.\exists y.(x \neq y))$$

- A fresh entity will always eventually be allocated:

$$GF(\exists x.x \text{ new})$$

- The number of entities that are continuously alive grows unboundedly:

$$G((F\exists x.x \text{ new}) \wedge \forall x.X(x \text{ alive}))$$

- Before x is deallocated, two new entities will be allocated:

$$x \text{ alive} \cup \exists y.(y \text{ new} \wedge (x \text{ alive} \cup \exists z.(z \text{ new} \wedge y \neq z \wedge x \text{ alive})))$$

- Every entity in the current state will be eventually deallocated:

$$\forall x.F(x \text{ dead})$$

- A deallocated entity cannot be reallocated (this will turn out to be a tautology):

$$x \text{ dead} \Rightarrow X(x \text{ dead})$$

- An entity that is identified now is no longer new in the next state (also a tautology):

$$X(x \text{ dead} \vee x \text{ old}).$$

2.2 Semantics

ATL formulae are essentially interpreted over infinite sequences of sets of entities.

Definition 2.2. An *allocational sequence* σ is an infinite sequence of sets of entities $E_0E_1E_2 \dots$ where $E_i \subseteq Ent$, for $i \in \mathbb{N}$.

Let $\sigma^i = E_iE_{i+1} \dots$. For given σ , E_i^σ denotes the set of entities in the i -th state of σ . The semantics of ATL-formulae is defined by satisfaction relation $\sigma, N, \theta \models \phi$ where σ is an allocational sequence, $N \subseteq E_0^\sigma$ is the set of entities that is initially new, and $\theta : LVar \rightarrow Ent$ is a partial valuation of the free variables in ϕ . Let N_i^σ denote the set of new entities in state i , defined by,

$$N_i^\sigma = \begin{cases} N & \text{if } i = 0 \\ E_i^\sigma \setminus E_{i-1}^\sigma & \text{otherwise.} \end{cases}$$

Similarly, let $\theta_i^\sigma : LVar \rightarrow Ent$ denote the valuation at state i , i.e., the valuation that is undefined for the variables that θ maps outside E_i^σ , and coincides with θ otherwise. Formally,

$$\theta_i^\sigma(x) = \begin{cases} \theta(x) & \text{if } \forall k \leq i : \theta(x) \in E_k^\sigma \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The previous definition imposes that if x denotes $e \in E_i$ then e must have been continuously alive (*non-resurrection* condition). The condition avoids that contradictions like $\exists x.X(x \text{ dead} \wedge Xx \text{ alive})$ are fulfilled. Note that once a logical variable is mapped to an entity, then this association remains along σ unless the entity dies, i.e., is deallocated.

Proposition 2.3. $\theta_{i+1}^\sigma = \theta_i^\sigma \upharpoonright E_{i+1}$.

The satisfaction relation \models is defined as follows:

$$\begin{aligned} \sigma, N, \theta \models x \text{ new} & \text{ iff } x \in \text{dom}(\theta) \text{ and } \theta(x) \in N \\ \sigma, N, \theta \models x = y & \text{ iff } x, y \in \text{dom}(\theta) \text{ and } \theta(x) = \theta(y) \\ \sigma, N, \theta \models \exists x.\phi & \text{ iff } \exists e \in E_0^\sigma : \sigma, N, \theta\{e/x\} \models \phi \\ \sigma, N, \theta \models \neg\phi & \text{ iff } \sigma, N, \theta \not\models \phi \\ \sigma, N, \theta \models \phi \vee \psi & \text{ iff either } \sigma, N, \theta \models \phi \text{ or } \sigma, N, \theta \models \psi \\ \sigma, N, \theta \models X\phi & \text{ iff } \sigma^1, N_1^\sigma, \theta_1^\sigma \models \phi \\ \sigma, N, \theta \models \phi \text{ U } \psi & \text{ iff } \exists i : (\sigma^i, N_i^\sigma, \theta_i^\sigma \models \psi \text{ and } \forall j < i : \sigma^j, N_j^\sigma, \theta_j^\sigma \models \phi). \end{aligned}$$

Here, $\theta\{e/x\}$ is defined as usual, i.e., $\theta\{e/x\}(x) = e$ and $\theta\{e/x\}(y) = \theta(y)$ for $y \neq x$.

Proposition 2.4. The following formulae are tautologies:

- $x = y \Rightarrow y = x$
- $x \text{ new} \Rightarrow x \text{ alive}$
- $x = y \Rightarrow X(x \text{ dead} \vee x = y)$
- $\forall x.x \text{ alive}$
- $X(x \text{ dead} \vee x \text{ old})$.
- $(x = y \wedge y = z) \Rightarrow x = z$
- $(x = y \wedge x \text{ new}) \Rightarrow y \text{ new}$
- $X(x = y) \Rightarrow x = y$
- $G(x \text{ dead}) \Rightarrow X(x \text{ dead})$

2.3 Folded allocational sequences

In order to come to a finite representation of allocational sequences there are several difficulties to overcome: the sequences are infinite themselves, and in general they range over an

infinite set of entities (i.e., $|\bigcup_{i \in \mathbb{N}} E_i| = \omega$). The former problem can be solved as usual, by generating allocational sequences as runs of a Büchi automaton; the latter problem requires a change in representation of the allocational sequences. This change is based on the following observations:

- ATL-formulae cannot address entities directly, but only through logical variables. The choice of representation for the entities is therefore irrelevant from the point of view of the ATL semantics.
- ATL-formulae allow the direct comparison of entities only within a state. The interpretation will not change if we allow to reallocate entities from a state to the next state, as long as this is done injectively so that distinct entities remain distinguished.

These considerations bring us to the following definition.

Definition 2.5. For $E, E' \subseteq Ent$,

- a *reallocation* λ from E to E' is a partial injective function $\lambda : E \rightarrow E'$.
- A *folded allocational sequence* is an infinite alternating sequence $E_0 \lambda_0 E_1 \lambda_1 \dots$, where λ_i is a reallocation from E_i to E_{i+1} for $i \geq 0$.

Thus, entity e is considered to be deallocated if $e \notin \text{dom}(\lambda)$. We write λ_i^σ for the reallocation function of σ in state i . Note that for folded allocational sequence σ with associated initial set N and valuation θ ,

$$N_i^\sigma = \begin{cases} N & \text{if } i = 0 \\ E_i^\sigma \setminus \text{cod}(\lambda_{i-1}^\sigma) & \text{otherwise.} \end{cases}$$

Similarly,

$$\theta_i^\sigma = \begin{cases} \theta & \text{if } i = 0 \\ \lambda_{i-1}^\sigma \circ \theta_{i-1}^\sigma & \text{otherwise.} \end{cases}$$

(Hence $\theta_i^\sigma = \lambda_{i-1}^\sigma \circ \dots \circ \lambda_0^\sigma \circ \theta$ for all $i \in \mathbb{N}$.) Using these adapted definitions of N and θ , the same definition of satisfaction relation \models holds for folded allocational sequences. In the following we indicate by \models_u and \models_f the semantics of ATL in the unfolded¹ case and in the folded case, respectively.

2.4 Relating unfolded and folded allocational sequences

For allocational sequence $\sigma = E_0 E_1 E_2 \dots$, let $id(\sigma)$ be a folded allocational sequence defined by $E_0 id_0 E_1 id_1 E_2 id_2 \dots$ where $id_i = id \upharpoonright (E_i \cap E_{i+1})$. We have the following straightforward fact:

Proposition 2.6. For any ATL-formula ϕ we have $\sigma, N, \theta \models_u \phi$ iff $id(\sigma), N, \theta \models_f \phi$.

¹In the following we will sometimes use *unfolded* allocational sequences in order to stress the difference w.r.t folded allocational sequences. When the context is clear and no ambiguities can arise, we will skip the adjectives folded or unfolded.

Neither folded nor unfolded allocational sequences are fully abstract with respect to the validity of ATL-formulas, as several of such sequences may satisfy the same formulae. We therefore consider folded allocational sequences *modulo isomorphism*. Folded allocational sequences σ_1 and σ_2 are isomorphic ($\sigma_1 \cong \sigma_2$) if there exists an indexed family of bijections $(h_i)_{i \in \mathbb{N}}$ with $h_i : E_i^{\sigma_1} \rightarrow E_i^{\sigma_2}$ such that $\lambda_i^{\sigma_2} \circ h_i = h_{i+1} \circ \lambda_i^{\sigma_1}$ (i.e., the h_i 's are consistent with the reallocations). Let $(\sigma_1, N_1, \theta_1) \cong (\sigma_2, N_2, \theta_2)$ if $\text{dom}(\theta_1) = \text{dom}(\theta_2)$, $\sigma_1 \cong \sigma_2$, and $N_2 = h_0(N_1)$ and $\theta_2 = h_0 \circ \theta_1$.

Proposition 2.7. For ATL-formula ϕ and folded allocational sequences σ, σ' :

$$(\sigma, N, \theta) \cong (\sigma', N', \theta') \Rightarrow (\sigma, N, \theta \models_f \phi \text{ iff } \sigma', N', \theta' \models_f \phi).$$

Proof. By a straightforward induction on the structure of ϕ . □

Unfolded and folded allocational sequences are related by \sqsubseteq^{fold} . Let $(\sigma_u, N_u, \theta_u) \sqsubseteq^{fold} (\sigma_f, N_f, \theta_f)$ iff $(id(\sigma_u), N_u, \theta_u) \cong (\sigma_f, N_f, \theta_f)$.

Proposition 2.8. For ATL-formula ϕ , folded allocational sequence σ_f and allocational sequence σ_u :

$$(\sigma_u, N_u, \theta_u) \sqsubseteq^{fold} (\sigma_f, N_f, \theta_f) \Rightarrow (\sigma_u, N_u, \theta_u \models_u \phi \text{ iff } \sigma_f, N_f, \theta_f \models_f \phi).$$

Proof. By definition $(\sigma_u, N_u, \theta_u) \sqsubseteq^{fold} (\sigma_f, N_f, \theta_f)$ iff $(id(\sigma_u), N_u, \theta_u) \cong (\sigma_f, N_f, \theta_f)$. By Proposition 2.7, $id(\sigma_u), N_u, \theta_u \models_f \phi$ iff $\sigma_f, N_f, \theta_f \models_f \phi$. From Proposition 2.6, it now follows that $\sigma_u, N_u, \theta_u \models_u \phi$ iff $\sigma_f, N_f, \theta_f \models_f \phi$. □

The following results show that allocational sequences and folded ones can both be used as models of ATL-formulae:

Proposition 2.9. For an arbitrary folded allocational sequence σ_f and unfolded allocational sequence σ_u :

1. For every $(\sigma_u, N_u, \theta_u)$ there exists a $(\sigma_f, N_f, \theta_f)$ such that $(\sigma_u, N_u, \theta_u) \sqsubseteq^{fold} (\sigma_f, N_f, \theta_f)$
2. For every $(\sigma_f, N_f, \theta_f)$ there exists a $(\sigma_u, N_u, \theta_u)$ such that $(\sigma_u, N_u, \theta_u) \sqsubseteq^{fold} (\sigma_f, N_f, \theta_f)$.

Proof. See Appendix A. □

Example 2.10. Let $\phi \equiv G(\exists x.x \text{ new})$ and $E_1 = \{e_1\}$, $E_{12} = \{e_1, e_2\}$, $E_{23} = \{e_2, e_3\}$. If the initial set of new entities $N = E_1$, the unfolded allocational sequence $\sigma = E_1(E_{12}E_{23})^\omega$ satisfies ϕ for any θ , whereas $\sigma' = E_1E_{12}^\omega$ does not, since after the second state, entities in E_{12} are continuously old. Let $\lambda_1 : E_1 \rightarrow E_{12}$ and $\lambda_2 : E_{12} \rightarrow E_{12}$ be two reallocations such that $\lambda_1(e_1) = e_2$, $\lambda_2(e_1) = e_2$ and $\lambda_2(e_2)$ is undefined. The folded allocational sequence $\sigma'_f = E_1\lambda_1(E_{12}\lambda_2)^\omega$ has the same sets of entities as σ' . Nevertheless, σ'_f satisfies ϕ . In fact, by the reallocation λ , the entity e_1 is new ($e_1 \notin \text{cod}(\lambda_1)$ and $e_1 \notin \text{cod}(\lambda_2)$) in every state. Moreover, in σ'_f the entity e_2 dies at every step while in σ' e_2 is continuously alive. Thus, the formula $\exists x.(XXx \text{ dead})$ is satisfied by σ'_f but not by σ' .

3 Allocational Büchi automata

In this section, we introduce two extensions of (generalised) Büchi automata [3]. Allocational Büchi automata (ABA) generate allocational sequences whereas high-level allocational Büchi automata (HABA) generate folded allocational sequences. Typically ABA are infinite state, whereas for the cases that we are interested in, the corresponding HABA is finite state. The HABA model is inspired by history-dependent (HD) automata [20]. The precise relationship between ABA and HABA is investigated below.

3.1 ABA

ABA are basically generalised Büchi automata where to each state a set of entities is associated. These entities, in turn, serve as valuation of logical (entity) variables.

Definition 3.1. An *Allocational Büchi Automaton* (ABA) \mathcal{A} is a tuple $\langle X, Q, E, \rightarrow, I, \mathcal{F} \rangle$, with

- $X \subseteq LVar$ a finite set of logical variables;
- Q a (possibly infinite) set of states;
- $E : Q \rightarrow 2^{Ent}$ a function yielding for each state q a finite set E_q of entities;
- $\rightarrow \subseteq Q \times Q$ a transition relation;
- $I : Q \rightarrow 2^{Ent} \times (X \rightarrow Ent)$ a partial function yielding for every *initial state* $q \in \text{dom}(I)$ an *initial valuation* (N, θ) , where $N \subseteq E_q$ is a finite set of entities, and $\theta : X \rightarrow Ent$ is a partial valuation of the variables in X ;
- $\mathcal{F} \subseteq 2^Q$ a set of sets of accept states.

Notational conventions: we write $(q, N, \theta) \in I$ for $I(q) = (N, \theta)$ and $q \rightarrow q'$ for $(q, q') \in \rightarrow$. We adopt the generalised Büchi acceptance condition, i.e, $\rho = q_0q_1q_2 \dots$ is a *run* of ABA \mathcal{A} if $q_i \rightarrow q_{i+1}$ for all $i \in \mathbb{N}$ and $|\{i | q_i \in F\}| = \omega$ for all $F \in \mathcal{F}$. Let $\text{runs}(\mathcal{A})$ denote the set of runs of \mathcal{A} . Run $\rho = q_0q_1q_2 \dots$ is said to *accept* the (unfolded) allocational sequence $\sigma = E_{q_0}E_{q_1}E_{q_2} \dots$. Let

$$\mathcal{L}(\mathcal{A}) = \{(\sigma, N, \theta) | \exists \rho = q_0q_1q_2 \dots \in \text{runs}(\mathcal{A}) : \rho \text{ accepts } \sigma \text{ and } I(q_0) = (N, \theta)\}.$$

Example 3.2. Fig. 1 depicts an infinite-state ABA \mathcal{A} for $X = \emptyset$ with initial state q_1 for which $I(q_1) = (\{e_1\}, \emptyset)$. In initial states, filled circles denote new entities. q_2, q_3, \dots are accept states (double circles). For simplicity, we assume $|\mathcal{F}| = 1$. \mathcal{A} accepts allocational sequences that start with a single (new) entity and then move to a state where an arbitrary number of new entities is created. From that point on, at every step a single new entity is added, therefore the formula $G(\exists x.x \text{ new})$ is satisfied by every triple $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{A})$. Note that entities are never deallocated, thus the formula $F(\exists x.(Fx \text{ dead}))$ is not satisfied by any of these triples in $\mathcal{L}(\mathcal{A})$.

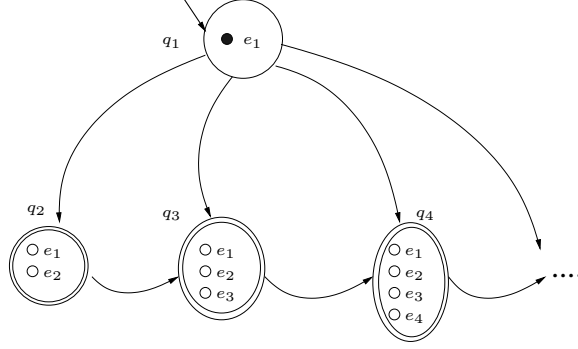


Figure 1: An example ABA

3.2 HABA

Let $\infty \notin Ent$ be a special, distinguished entity, called *black hole*. Its role will become clear later on. We denote $E^\infty = E \cup \{\infty\}$ for arbitrary $E \subseteq Ent$.

Definition 3.3. For $E, E_1 \subseteq Ent$, an ∞ -*reallocation* is a partial function $\lambda : E^\infty \rightarrow E_1^\infty$ such that

- $\lambda(e) = \lambda(e') \neq \infty \Rightarrow e = e'$ for all $e, e' \in E$ and
- $\infty \in \text{dom}(\lambda) \Rightarrow \lambda(\infty) = \infty$.

That is, λ is injective when mapping away from ∞ and preserves ∞ .

Definition 3.4. A *High-level ABA* (HABA) \mathcal{H} is a tuple $\langle X, Q, E, \rightarrow, I, \mathcal{F} \rangle$ with X, Q, I, \mathcal{F} as in Def. 3.1, and

- $E : Q \rightarrow 2^{Ent} \times \mathbb{B}$, a function that associates to each state $q \in Q$ a finite set E_q of entities and a predicate B_q expressing that there is an unbounded number of entities in q not explicitly modelled by E_q ;
- $\rightarrow \subseteq Q \times (Ent^\infty \rightarrow Ent^\infty) \times Q$, such that for $q \rightarrow_\lambda q'$, λ is an ∞ -reallocation from E_q^∞ to $E_{q'}^\infty$ with
 - (i) $\infty \in \text{dom}(\lambda)$ iff $E_q = (E, \text{ff})$ and $E_{q'} = (E', \text{ff})$, and
 - (ii) $\infty \in \text{cod}(\lambda) \Rightarrow E_{q'} = (E', \text{ff})$.

A HABA is a *symbolic* representation of a (possibly infinite) ABA. Predicate B_q holds in state q if and only if the number of entities in q is bounded. An unbounded state q (denoted $\lfloor q \rfloor$), possesses the distinguished entity ∞ that represents all entities that may be added to q (*imploded entities*). High-level state q thus represents all possible concrete states obtained from q by adding a finite number of entities to E_q . If a transition to state q' maps (implodes) entities into the black hole ∞ , these entities cannot be distinguished anymore from there on. Moreover, if $q \rightarrow_\lambda q'$, entities in the black hole are either preserved (if $\lfloor q' \rfloor$), or are destroyed (if $\lceil q' \rceil$). The black hole thus allows to abstract from the identity of entities when these are not relevant anymore. Note that $\infty \notin Ent$ implies $\infty \notin \text{cod}(\theta)$ for all $(q, N, \theta) \in I$.

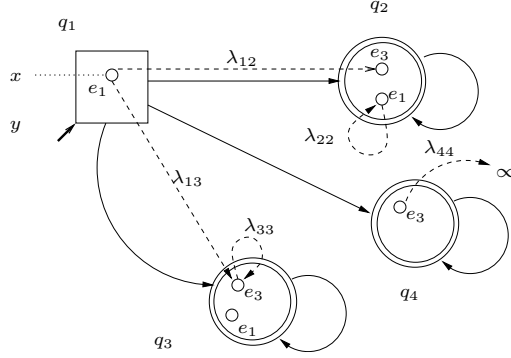


Figure 2: An example HABA

Example 3.5. Figure 2 depicts a HABA with $X = \{x, y\}$. Squares denote bounded states, (large) circles denote unbounded states, small circles denote entities, and accept states have a double boundary. Dashed arrows indicate ∞ -reallocations. In initial states, dotted lines represent θ (for a complete summary of the visual notation for HABAs, see Figure 3). In q_1 , variable x denotes (old) entity e_1 , while y is undefined. Entity e_3 in state q_2 represents the same entity as e_1 in q_1 , while e_1 (in q_2) represents a new entity.

Run $q_1 \lambda_{12} (q_2 \lambda_{22})^\omega$ generates sequences where the initial entity dies after the second state, while the new entity created in the second state will be alive forever. Run $q_1 \lambda_{13} (q_3 \lambda_{33})^\omega$ generates sequences where the initial entity will be alive forever, and in each state a new entity is created. This new entity will die in the next state. Finally, run $q_1 \lambda_{14} (q_4 \lambda_{44})^\omega$ (the reallocation λ_{14} is not depicted since it is empty) generates sequences where the entity in the initial state dies immediately. Once q_4 is reached, a new entity e_3 is created at every step, and in this run thus the number of entities grows unboundedly.

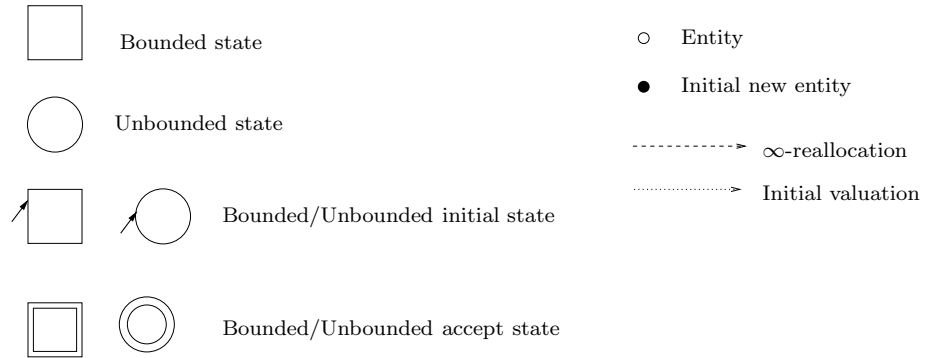


Figure 3: Visual notation for HABAs

In the above example, we remarked that the runs of a HABA generate folded allocational sequences. We make this concept precise with the following:

Definition 3.6. A run $\rho = q_0 \lambda_0 q_1 \lambda_1 \dots$ of HABA $\mathcal{H} = \langle X, Q, E, \rightarrow, I, \mathcal{F} \rangle$ generates an allocation triple (σ, N, θ) , where $\sigma = E_0 \lambda_0^\sigma E_1 \lambda_1^\sigma \dots$ is a folded allocational sequence, if there is a *generator*, i.e., a family of functions $\phi_i : E_i \rightarrow E_{q_i}^\infty$ satisfying for all $i \geq 0$:

1. $\forall e, e' \in E_i. (\phi_i(e) = \phi_i(e') \neq \infty \Rightarrow e = e')$
2. $E_{q_i} \subseteq \text{cod}(\phi_i)$
3. $[q_i] \Rightarrow \infty \notin \text{cod}(\phi_i)$
4. $\lambda_i \circ \phi_i = \phi_{i+1} \circ \lambda_i^\sigma$
5. $\forall e \in E_{i+1}. (\phi_{i+1}(e) = \infty \Rightarrow e \in \text{cod}(\lambda_i^\sigma))$
6. $I(q_0) = (\phi_0(N), \phi_0 \circ \theta)$

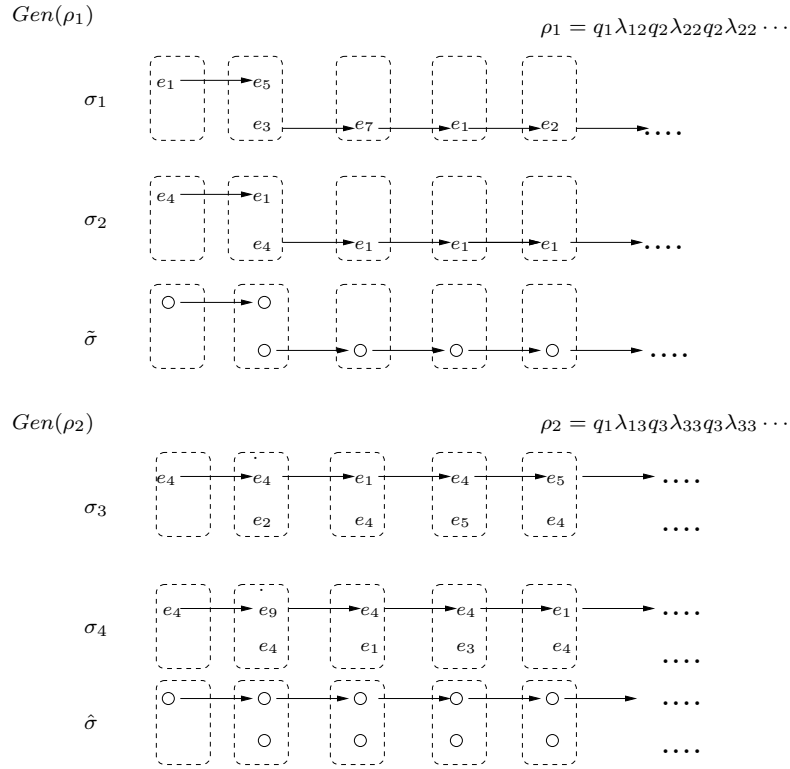


Figure 4: Example folded allocational sequences generated by runs ρ_1 and ρ_2 of the HABA of Fig. 2.

Condition 1 expresses that ϕ_i is injective except for entities imploded (mapped) onto ∞ . Condition 2 says that every entity in state q_i represents an entity of the state E_i of the allocational sequence. Condition 3 ensures that a bounded state does not generate states of the allocational sequence with entities corresponding to imploded ones. Condition 4 ensures that λ_i^σ is consistent to λ_i in the reallocation of corresponding entities. Condition 5 ensures that the number of new entities in the allocational sequence is the same as the number of new entities in the state of the model. The last condition relates the initial valuation of the allocation triple (σ, N, θ) to the initial valuation of HABA \mathcal{H} .

Runs of a HABA are defined in the same way as for ABA. Let $\text{runs}(\mathcal{H})$ denote the set of runs of \mathcal{H} and $\mathcal{L}(\mathcal{H}) = \{(\sigma, N, \theta) \mid \exists \rho \in \text{runs}(\mathcal{H}) : \rho \text{ generates } (\sigma, N, \theta)\}$.

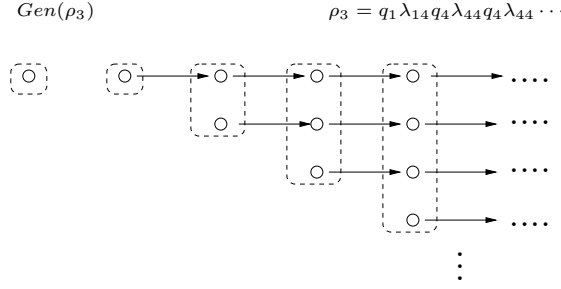


Figure 5: Folded allocational sequences generated by run ρ_3 of the HABA of Fig. 2.

Example 3.7. Figure 4 shows some sequences generated by the HABA in Figure 2. Dashed boxes represent states of the sequences. In particular let $\rho_1 = q_1\lambda_{12}q_2\lambda_{22}q_2\lambda_{22}\dots$ and $\rho_2 = q_1\lambda_{13}q_3\lambda_{33}q_3\lambda_{33}\dots$ then $\sigma_1, \sigma_2 \in \text{Gen}(\rho_1)$ and $\sigma_3, \sigma_4 \in \text{Gen}(\rho_2)$. Again, note how the identity of the entities is not relevant and in fact, in general, the set $\text{Gen}(\rho_1)$ contains all the sequences isomorphic to σ_1 (and/or σ_2) and $\text{Gen}(\rho_2)$ contains all the sequences isomorphic to σ_3 (and/or σ_4).

We can abstract from the identity of the entities and represent sequences by anonymous entities depicted just by a circle. This shows the “general pattern” followed by all isomorphic sequences. For example in Figure 4, $\text{Gen}(\rho_1)$ follows the general pattern represented by $\tilde{\sigma}$ while those in $\text{Gen}(\rho_2)$ follow the pattern of $\tilde{\delta}$.

Figure 5 depicts the pattern of the sequences in $\text{Gen}(\rho_3)$ where $\rho_3 = q_1\lambda_{14}q_4\lambda_{44}q_4\lambda_{44}\dots$. The number of entities grows at every step.

Finally, assume that q_1 would have been unbounded. Then, $\text{Gen}(\rho_1)$, $\text{Gen}(\rho_2)$ and $\text{Gen}(\rho_3)$ would contain also those sequences with an arbitrary number of (additional) imploded entities. This situation is represented in Figure 6 where, in order to stress the difference w.r.t. the bounded case, we have repeated $\tilde{\sigma}$ from Figure 4. The two sequences σ_1 and σ_2 in the bottom (of Fig. 6) are obtained by adding to $E_0^{\tilde{\sigma}}$ two and three imploded entities (here depicted as filled circles) respectively. In this example, imploded entities are preserved by every transition because q_2 is unbounded (recall condition $\lambda(\infty) = \infty$ in the definition of ∞ -reallocation and condition on HABA transitions).

3.3 The duality between ABA and HABA

The relationship between ABAs and HABAs strongly depends on the relation between unfolded and folded allocational sequences discussed in Section 2. More precisely, we will establish a connection between a HABA \mathcal{H} and a class of particular ABAs, called the expansions of \mathcal{H} , whose elements accept $\mathcal{L}(\mathcal{H})$ up to isomorphism.

Definition 3.8. For HABA \mathcal{H} and ABA \mathcal{A} , let $\psi : Q_{\mathcal{A}} \rightarrow Q_{\mathcal{H}}$ be surjective, and $(\phi_q)_{q \in Q_{\mathcal{A}}}$ be a family of functions $\phi_q : E_q \rightarrow E_{\psi(q)}^{\infty}$. Then:

- \mathcal{A} -transition $q_1 \rightarrow q_2$ expands \mathcal{H} -transition $q'_1 \rightarrow_{\lambda} q'_2$ if $q'_1 = \psi(q_1)$, $q'_2 = \psi(q_2)$ and
 - (i) $\lambda \circ \phi_{q_1} = \phi_{q_2} \upharpoonright (E_{q_1} \cap E_{q_2})$ and
 - (ii) $|E_{q'_2} \setminus \text{cod}(\lambda)| = |E_{q_2} \setminus E_{q_1}|$.

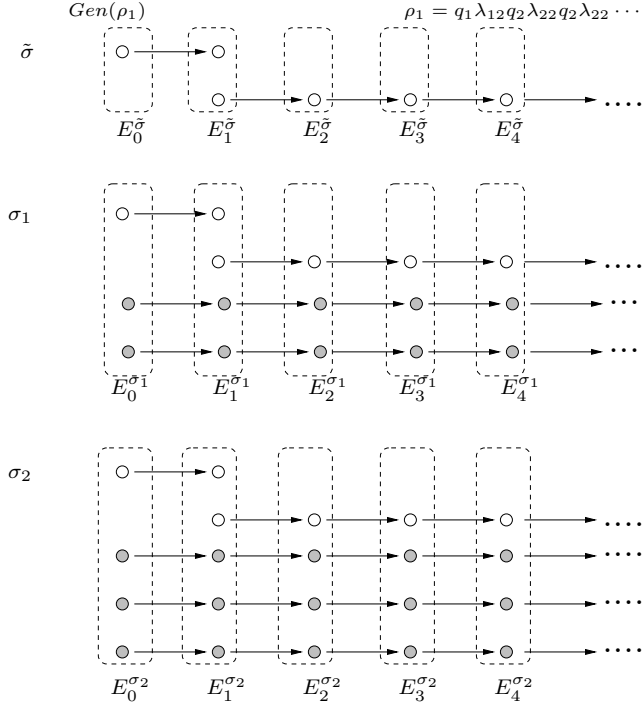


Figure 6: Folded allocational sequences generated for by HABA of Fig. 2 if $[q_1]$.

- \mathcal{A} is an *expansion* of \mathcal{H} if the following conditions are satisfied:
 1. $\forall e, e' \in E_q. (\phi_q(e) = \phi_q(e') \neq \infty \Rightarrow e = e')$;
 2. $E_{\psi(q)} \subseteq \text{cod}(\phi_q)$;
 3. $[\psi(q)] \Rightarrow \infty \notin \text{cod}(\phi_q)$;
 4. for all $q_1 \in Q_{\mathcal{A}}$,
 - a) for all $\psi(q_1) \rightarrow_{\lambda} q'_2$ there exists $q_1 \rightarrow q_2$ that expands $\psi(q_1) \rightarrow_{\lambda} q'_2$
 - b) for all $q_1 \rightarrow q_2$ there exists λ such that $q_1 \rightarrow q_2$ expands $\psi(q_1) \rightarrow_{\lambda} \psi(q_2)$;
 5. $I_{\mathcal{A}} : q \mapsto \begin{cases} (\phi_q^{-1}(N), \phi_q^{-1} \circ \theta) & \text{if } q' = \psi(q) \text{ and } I_{\mathcal{H}}(q') = (N, \theta) \\ \text{undefined} & \text{otherwise} \end{cases}$
 6. $\mathcal{F}_{\mathcal{A}} = \{\{\psi(q)|q \in F\}|F \in \mathcal{F}_{\mathcal{H}}\}$.

The first three conditions are taken from Def. 3.6. Intuitively, these force the number of entities in an expanded state to exceed the number of entities of the original state, and require equality if the original state is bounded. Note that transition $q_1 \rightarrow q_2$ in the ABA must preserve the reallocation of $q'_1 \rightarrow_{\lambda} q'_2$ (condition (i)) as well as the number of new entities (condition (ii)).

Some notions for folded and unfolded allocational sequences are lifted to accepted languages in the following way. For HABAs \mathcal{H}_1 and \mathcal{H}_2 , let $\mathcal{L}(\mathcal{H}_1) \cong \mathcal{L}(\mathcal{H}_2)$ iff for all $(\sigma_1, N_1, \theta_1) \in \mathcal{L}(\mathcal{H}_1)$ there exists a $(\sigma_2, N_2, \theta_2) \in \mathcal{L}(\mathcal{H}_2)$ such that $(\sigma_1, N_1, \theta_1) \cong (\sigma_2, N_2, \theta_2)$ and vice versa. For ABA \mathcal{A} accepting $\mathcal{L}(\mathcal{A})$, let $id(\mathcal{L}(\mathcal{A})) = \{(id(\sigma), N, \theta) | (\sigma, N, \theta) \in \mathcal{L}(\mathcal{A})\}$. Note that $id(\mathcal{L}(\mathcal{A}))$ is the folded version of the unfolded language $\mathcal{L}(\mathcal{A})$. For HABA \mathcal{H} and ABA

\mathcal{A} let $\mathcal{L}(\mathcal{A}) \sqsubseteq^{fold} \mathcal{L}(\mathcal{H})$ iff for all $(\sigma_u, N_u, \theta_u) \in \mathcal{L}(\mathcal{A})$ there exists $(\sigma_f, N_f, \theta_f) \in \mathcal{L}(\mathcal{H})$ such that $(\sigma_u, N_u, \theta_u) \sqsubseteq^{fold} (\sigma_f, N_f, \theta_f)$. Similarly, $\mathcal{L}(\mathcal{A}) \sqsupseteq^{fold} \mathcal{L}(\mathcal{H})$ iff for all $(\sigma_f, N_f, \theta_f) \in \mathcal{L}(\mathcal{H})$ there exists $(\sigma_u, N_u, \theta_u) \in \mathcal{L}(\mathcal{A})$ such that $(\sigma_u, N_u, \theta_u) \sqsubseteq^{fold} (\sigma_f, N_f, \theta_f)$. Then:

Lemma 3.9. For HABA \mathcal{H} and any expansion $Exp(\mathcal{H})$ of \mathcal{H} :

- (a) $\mathcal{L}(Exp(\mathcal{H})) \sqsupseteq^{fold} \mathcal{L}(\mathcal{H})$ and
- (b) $\mathcal{L}(Exp(\mathcal{H})) \sqsubseteq^{fold} \mathcal{L}(\mathcal{H})$.

Proof. See Appendix B. □

Hence, $\mathcal{L}(Exp(\mathcal{H}))$ corresponds precisely to the unfolded version of $\mathcal{L}(\mathcal{H})$. A consequence of the previous lemma, as well as an alternative way to express it, is stated by:

Theorem 3.10. For any HABA \mathcal{H} and expansion $Exp(\mathcal{H})$ of \mathcal{H} : $\mathcal{L}(\mathcal{H}) \cong id(\mathcal{L}(Exp(\mathcal{H})))$.

Proof. See Appendix B. □

Definition 3.11. Given a HABA \mathcal{H} and an ATL-formula ϕ we say that

- ϕ is \mathcal{H} -satisfiable if there exists $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{H})$ such that $\sigma, N, \theta \models \phi$;
- ϕ is \mathcal{H} -valid if for all $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{H})$: $\sigma, N, \theta \models \phi$.

For a ABA \mathcal{A} an equivalent definition for \mathcal{A} -satisfiability and \mathcal{A} -validity can be given in precisely the same way. From the previous definition it follows that ϕ is \mathcal{H} -valid (\mathcal{A} -valid) if and only if $\neg\phi$ is not \mathcal{H} -satisfiable (\mathcal{A} -satisfiable).

As stated in the following corollary, from Theorem 3.10 it follows that a HABA \mathcal{H} and its expansions satisfy the same set of ATL formulae.

Corollary 3.12. For any ATL-formula ϕ , HABA \mathcal{H} and expansion $Exp(\mathcal{H})$ of \mathcal{H} : ϕ is \mathcal{H} -satisfiable if and only if ϕ is $Exp(\mathcal{H})$ -satisfiable.

Figure 7 shows an example of a HABA and one of its (infinite number of) infinite expansion.

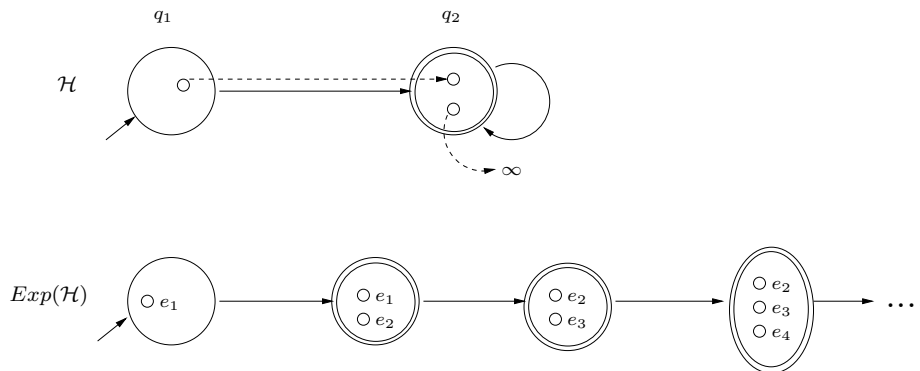


Figure 7: A HABA and one of its expansions.

4 Programming allocation and deallocation

This section introduces a simple programming language \mathcal{L} capturing the essence of allocation and deallocation. It is used for providing an intuition about the setup and the sort of behaviour that can be modelled by HABA. Two semantics for \mathcal{L} are defined, a concrete semantics with ABA as underlying model, and a symbolic semantics using HABA. The relation between these semantics is shown to correspond to expansion in the sense of Def. 3.8.

4.1 Syntax

For $PVar$ a set of program variables with $v, v_i \in PVar$ and $PVar \cap LVar = \emptyset$, the set of statements of \mathcal{L} is given by:

$$\begin{aligned}
 (p \in) \mathcal{L} & ::= \text{decl } v_1, \dots, v_n : (s_1 \parallel \dots \parallel s_k) \\
 (s \in) Stat & ::= \text{new}(v) \mid \text{del}(v) \mid v := v \mid \text{skip} \mid s; s \mid \text{if } b \text{ then } s \text{ else } s \text{ fi} \mid \text{while } b \text{ do } s \text{ od} \\
 (b \in) Bexp & ::= v = v \mid b \vee b \mid \neg b
 \end{aligned}$$

A program p is thus a parallel composition of a finite number of statements preceded by the declaration of a finite number of global variables.

Informal semantics of \mathcal{L} $\text{new}(v)$ creates (i.e., allocates) a new entity that will be referred to by the program variable v . The old value of v is lost. Thus, if v is the only variable that refers to entity e , say, then after the execution of $\text{new}(v)$, e cannot be referenced anymore. In particular, e cannot be deallocated anymore. In other words, there is no automatic garbage collection. $\text{del}(v)$ destroys (i.e., deallocates) the entity associated to v , and makes v undefined. The assignment $v := w$ passes the reference held by w (if any) to v . Again, the entity v was referring to might become unreferenced (for ever). Sequential composition, while loop, skip, and conditional statement have the standard interpretation. For the sake of simplicity, new and del create and destroy, respectively, a single entity only; generalisations in which several entities are considered simultaneously can be added in a straightforward manner.

Example 4.1. The following program PC is an implementation of a producer/consumer system.

$$\begin{aligned}
 PC & \equiv \text{decl } v_1, v_2, w : (Prod \parallel B_2 \parallel Con) \text{ where} \\
 Prod & \equiv \text{while tt do} \\
 & \quad \text{if } (v_1 \text{ dead}) \text{ then } \text{new}(v_1) \text{ else skip fi} \\
 & \quad \text{od} \\
 B_2 & \equiv \text{while tt do} \\
 & \quad \text{if } (v_1 \text{ alive} \wedge v_2 \text{ dead}) \text{ then } w := v_1; v_1 := v_2; v_2 := w \text{ else skip fi} \\
 & \quad \text{od} \\
 Con & \equiv \text{while tt do} \\
 & \quad \text{if } (v_2 \text{ alive}) \text{ then } \text{del}(v_2) \text{ else skip fi} \\
 & \quad \text{od}
 \end{aligned}$$

The component B_2 implements a process handling a two-place buffer consisting of the variable v_1 (first position) and v_2 (second position). Note that in the if-then-else-fi statement, using $v_2 := v_1; \text{del}(v_1)$ would be wrong as the entity referenced by v_1 and v_2 would be deallocated.

Whereas, $v_1 := v_2$ deletes only the reference to v_1 entity that can be then moved to v_2 via w . The producer *Prod* produces a new entity when the first place of the buffer v_1 is empty (i.e. v_1 is dead). Process B_2 shifts the entity in the second position v_2 if this is empty. The variable w is only used as a temporary variable in order to swap the value of v_1 and v_2 . Finally an entity in the second position of the buffer is consumed by the consumer *Con*. Some examples properties expressible in ATL that may possibly be satisfied by the program *PC* are:

- An entity in the buffer is not consumed before the insertion of another one (this implies also that buffer is never empty): $\mathbf{G}\forall x.(x \text{ alive } \mathbf{U} (\exists y.x \neq y))$.
- The buffer will be always eventually empty: $\mathbf{GF}(\forall x.x \text{ dead})$.
- The buffer will be always eventually full: $\mathbf{GF}(\forall x.\forall y.\forall z.(x = y \vee x = z \vee y = z))$.
- A produced entity is always eventually consumed $\mathbf{G}(x \text{ alive} \Rightarrow \mathbf{F}x \text{ dead})$.
- Entities are always consumed in the same order they are produced:

$$\mathbf{G}(\forall x.(x = x \wedge \mathbf{XF}(\exists y.x \neq y \Rightarrow y \text{ alive } \mathbf{U} x \text{ dead})).$$

Example 4.2. The following program, where $g(i) = (i+1) \bmod 4$, models the implementation of a naive solution to the dining philosopher problem:

```

DPhil  ≡ decl  $v_1, v_2, v_3, v_4 : (Ph_1 \parallel Ph_2 \parallel Ph_3 \parallel Ph_4)$  where
Phi   ≡ while tt do
           if ( $v_i$  alive  $\wedge v_{g(i)}$  alive) then
             del( $v_i$ ); del( $v_{g(i)}$ ); new( $v_i$ ); new( $v_{g(i)}$ )
           else
             skip
           fi
         od

```

The variables v_i and $v_{g(i)}$ represent the left and the right chopstick of philosopher Ph_i , respectively. If v_i and $v_{g(i)}$ are defined, then the chopsticks are on the table. Taking the chopsticks from the table is represented by destroying the corresponding entities, while putting the chopsticks back on the table is modelled by creating new entities. Some ATL properties that may possibly be satisfied by this program, are:

- Eventually two philosophers will eat at the same time:

$$\mathbf{F}(\forall x.x \text{ dead}).$$

- There is always eventually a moment in which every philosopher is thinking at the same time:

$$\mathbf{GF}(\exists x.\exists y.\exists z.\exists w.(x \neq y \wedge x \neq z \wedge x \neq w \wedge y \neq z \wedge y \neq w \wedge z \neq w)).$$

- There exists at least two philosophers that are never able to eat (an unfair computation):

$$\mathbf{G}(\forall x.\forall y.\forall z.(x = y \vee x = z \vee y = z)).$$

- Among the philosophers there exists a greedy impostor who always eats and never thinks (again an unfair computation):

$$\text{G}(\forall x.\forall y.\forall z.(x \text{ old} \wedge y \text{ old} \wedge z \text{ old} \wedge (x = y \vee x = z \vee y = z))).$$

- Eventually there is only one chopstick on the table (an inconsistency):

$$\text{F}(\forall x.\forall y.(x = y)).$$

Although some interesting problems can be programmed, it is obvious that \mathcal{L} is rather simple. Other constructs like `wait` or some syntactic sugar like `until` may be easily included, without extending the language in an essential way.

4.2 Concrete semantics

A concrete semantics of our example language is given in terms of ABA. Let Par denote the compound statements, i.e., $r \in Par ::= s \mid r \parallel s$. The concrete semantics of $p = \text{decl } v_1, \dots, v_n : (s_1 \parallel \dots \parallel s_k)$ is the ABA $\mathcal{A}_p = \langle \emptyset, Q, E, \rightarrow, I, \mathcal{F} \rangle$ where

- $Q \subseteq Par \times 2^{Ent} \times (PVar \rightarrow Ent)$, where for state $q = (r, E_q, \gamma_q) \in Q$, r is the compound statement to be executed, E_q is the set of entities alive and γ_q maps program variables to E_q (with $\text{dom}(\gamma_q) = \{v_1, \dots, v_n\}$; if γ_q is undefined on v we write $\gamma_q(v) = \perp$);
- $E(r, E', \gamma) = E'$;
- $\rightarrow \subseteq Q \times Q$ is the smallest relation satisfying the rules in Table 1;
- $\text{dom}(I) = \{(s_1 \parallel \dots \parallel s_k, \emptyset, \emptyset)\}$ and $I(s_1 \parallel \dots \parallel s_k, \emptyset, \emptyset) = (\emptyset, \emptyset)$;
- $\mathcal{F} = \{ \{ (s'_1 \parallel \dots \parallel s'_k, E, \gamma) \in Q \mid s'_i = \text{skip} \vee s'_i = \text{while } b \text{ do } s \text{ od}; s'' \} \mid 0 < i \leq k \}$.

A few remarks are in order. \mathcal{A}_p has a single initial state $s_1 \parallel \dots \parallel s_k$. The set of accept states for the i -th sequential component consists of all states in which the component has either terminated ($s_i = \text{skip}$) or is processing a loop (which could be infinite). The semantics of the boolean expressions is given by the function $\mathcal{V} : Bexp \times (PVar \rightarrow Ent) \rightarrow \mathbb{B}$ defined by

$$\begin{aligned} \mathcal{V}(v = w)(\gamma) &= \begin{cases} \text{tt} & \text{if } v, w \in \text{dom}(\gamma) \text{ and } \gamma(v) = \gamma(w) \\ \text{ff} & \text{otherwise} \end{cases} \\ \mathcal{V}(b_1 \vee b_2)(\gamma) &= \mathcal{V}(b_1)(\gamma) \vee \mathcal{V}(b_2)(\gamma) \\ \mathcal{V}(\neg b)(\gamma) &= \neg \mathcal{V}(b)(\gamma). \end{aligned}$$

We assume w.l.o.g. that the set Ent is totally ordered; this is convenient for selecting a fresh entity in a deterministic way (cf. the rule **NEW-conc** in Table 1). Some brief explanation of the rules of the concrete semantics follows:

- (**ASGN-conc**) An assignment $v := w$ is performed by changing the reference of the variable v with $\gamma(w)$. After the execution, the assignment statement is replaced by `skip` that is either consumed in the context of a sequential composition by rule (**SEQ₂-conc**) or is blocked. This general pattern is followed also by rules (**NEW-conc**) and (**DEL-conc**). Note that there does not exist a specific rule for `skip`.

Table 1: Operational rules for concrete semantics

(ASGN-conc)	$\frac{}{v := w, E, \gamma \rightarrow \text{skip}, E, \gamma\{\gamma(w)/v\}}$
(NEW-conc)	$\frac{}{\text{new}(v), E, \gamma \rightarrow \text{skip}, E \cup \{e\}, \gamma\{e/v\}} \quad e = \min(\text{Ent} \setminus E)$
(DEL-conc)	$\frac{}{\text{del}(v), E, \gamma \rightarrow \text{skip}, E \setminus \{\gamma(v)\}, \gamma\{\perp/v'\}} \quad v' \in \gamma^{-1}(\gamma(v))$
(IF ₁ -conc)	$\frac{\mathcal{V}(b)(\gamma)}{\text{if } b \text{ then } s_1 \text{ else } s_2 \text{ fi}, E, \gamma \rightarrow s_1, E, \gamma}$
(IF ₂ -conc)	$\frac{\neg \mathcal{V}(b)(\gamma)}{\text{if } b \text{ then } s_1 \text{ else } s_2 \text{ fi}, E, \gamma \rightarrow s_2, E, \gamma}$
(WHILE-conc)	$\frac{}{\text{while } b \text{ do } s \text{ od}, E, \gamma \rightarrow \text{if } b \text{ then } s; \text{ while } b \text{ do } s \text{ od else skip fi}, E, \gamma}$
(SEQ ₁ -conc)	$\frac{s_1, E, \gamma \rightarrow s'_1, E', \gamma'}{s_1; s_2, E, \gamma \rightarrow s'_1; s_2, E', \gamma'}$
(SEQ ₂ -conc)	$\frac{}{\text{skip}; s_2, E, \gamma \rightarrow s_2, E, \gamma}$
(PAR ₁ -conc)	$\frac{1 \leq j \leq k \wedge s_j, E, \gamma \rightarrow s'_j, E', \gamma'}{s_1 \parallel \dots \parallel s_j \parallel \dots \parallel s_k, E, \gamma \rightarrow s_1 \parallel \dots \parallel s'_j \parallel \dots \parallel s_k, E', \gamma'}$
(PAR ₂ -conc)	$\frac{}{\text{skip} \parallel \dots \parallel \text{skip}, E, \gamma \rightarrow \text{skip} \parallel \dots \parallel \text{skip}, E, \gamma}$

- (NEW-conc) The reference of the first (fresh) entity e available from Ent according to the total order is assigned to v .
- (DEL-conc) Entity $\gamma(v)$ is deallocated and every reference to this entity is cancelled.
- (IF₁-conc)/(IF₂-conc)/(WHILE-conc) Straightforward.
- (SEQ₁-conc)/(SEQ₂-conc) In a sequential composition, when the first statement is reduced to a skip statement, it is consumed.
- (PAR₁-conc) If one of the components of the compound statement performs a step, the whole compound statement can do so.
- (PAR₂-conc) A self-loop in an accept state with a terminated compound statement ensures that, in a run, it is visited infinitely many times.

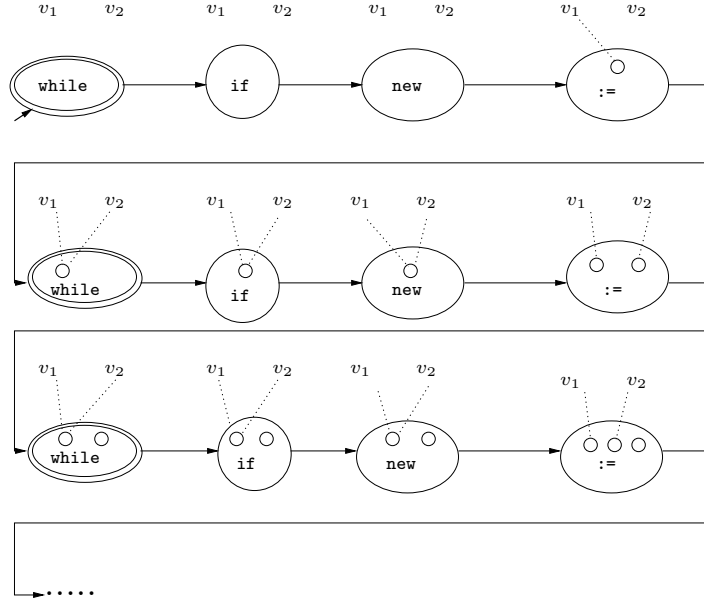


Figure 8: Concrete semantics of $\text{decl } v_1, v_2 : (\text{while tt do new}(v_1); v_2 := v_1 \text{ od})$.

Example 4.3. Consider the following program:

$$p \equiv \text{decl } v_1, v_2 : (\text{while tt do new}(v_1); v_2 := v_1 \text{ od}).$$

For the sake of abbreviation, we write

- **while** for “while tt do new(v_1); $v_2 := v_1$ od”
- **if** for “if tt then new(v_1); $v_2 := v_1$; while else skip fi”
- **new** for “new(v_1); $v_2 := v_1$; while”
- **:=** for “ $v_2 := v_1$; while”.

The ABA in Figure 8 represents the concrete semantics \mathcal{A}_p of p . \mathcal{A}_p is infinite since in every iteration a new entity is created.

4.3 Symbolic semantics

The semantics defined in the previous section has the disadvantage that models may become infinite due to an unbounded number of entity creations. In order to circumvent this problem we define a symbolic semantics of \mathcal{L} in terms of HABA, that (later on) will be shown to be equivalent. The main distinction with the concrete semantics is the treatment of the association of program variables to entities. Here instead, entities are represented by a partial partition of a subset of $PVar$, i.e., the set E of entities is of the form $\{X_1, \dots, X_n\}$ with $X_i \subseteq PVar$ and $X_i \cap X_j = \emptyset$ (for $i \neq j$). Note that we do not require $\bigcup_i X_i = PVar$ which would make it a full partitioning. Variable v is defined if and only if $v \in X_i$ for some i . Then, v refers to the entity represented by the set X_i . Otherwise, v is undefined. Using this approach, there is no need to represent (in a state) a mapping from the set of program variables onto the entities. The symbolic semantics of $p = \text{decl } v_1, \dots, v_n : (s_1 \parallel \dots \parallel s_k)$ is the HABA $\mathcal{H}_p = \langle \emptyset, Q, E, \rightarrow, I, \mathcal{F} \rangle$ where

- $Q \subseteq Par \times 2^{PVar}$, i.e., a state $q = (r, E)$ consists of a compound statement and a set of entities; we have $[q]$ iff $\emptyset \in E$ (i.e., we represent the black hole by \emptyset).
- $E(r, E') = E' \setminus \{\emptyset\}$;
- \rightarrow is the smallest relation defined by the rules in Table 2 such that for $r, E \rightarrow_\lambda r', E'$ we have $\emptyset \in E \Rightarrow \emptyset \in \text{dom}(\lambda)$.

and I and \mathcal{F} are defined in the same way as for the concrete semantics. The condition on \emptyset in the definition of \rightarrow can be seen as a kind of “preservation law” of the black hole. In fact, once a state implodes into an unbounded one, the black hole generated by this implosion will last forever. Note that in Def. 3.4 this is not always the case. Whenever entity X_i is not referenced by any program variable, the state will become unbounded. Entity X_i will then be mapped by λ onto \emptyset (recall that in the special case of \mathcal{H}_p we represent ∞ by \emptyset), which can be viewed as a “black hole” collecting every non-referenced entity. These entities share the property that they cannot be deallocated anymore, thus they will have exactly the same future, namely they will be “floating” in the black hole *ad infinitum*.

Some explanations on the rules of the symbolic semantics are in order:

- (NEW-sym) If v is the only variable having a reference to an entity X_i (i.e., $X_i = \{v\}$), the state becomes unbounded (if this is not already the case) and the black hole \emptyset implodes X_i since it cannot be referred to anymore. In this case, $E' = E \cup \{\emptyset\}$, i.e., the sets E and E' have the same entities. However, X_i represents a new entity in the target state since $X_i \notin \text{cod}(\lambda)$.

If v is either undefined or there exists another variable denoting v 's entity, a new entity $\{v\}$ is created. λ maps every entity into itself.

- (ASGN-sym) If v is defined but is the only variable that has a reference to its entity, the assignment causes the loss of the reference of the entity denoted by v . Therefore, this entity is imploded onto \emptyset . If w is undefined also v becomes undefined. Regardless whether the state is bounded or not, after the transition the state becomes unbounded because $\emptyset \in E'$.

If there is another variable denoting v 's entity or v is undefined then v 's reference is changed.

Table 2: Operational rules for symbolic semantics

(ASGN-sym)	$\frac{}{v := w, E \rightarrow_\lambda \text{skip}, \{X_i \setminus \{v\} w \notin X_i\} \cup \{X_i \cup \{v\} w \in X_i\}} \quad \lambda : X_i \mapsto \begin{cases} X_i \setminus \{v\} & \text{if } w \notin X_i \\ X_i \cup \{v\} & \text{otherwise} \end{cases}$
(NEW-sym)	$\frac{}{\text{new}(v), E \rightarrow_\lambda \text{skip}, \{X_i \setminus \{v\} X_i \in E\} \cup \{\{v\}\}} \quad \lambda(X_i) = X_i \setminus \{v\}$
(DEL-sym)	$\frac{v \in X_i}{\text{del}(v), E \rightarrow_\lambda \text{skip}, (E \setminus \{X_i\})} \quad \lambda : X_j \mapsto \begin{cases} X_j & \text{if } j \neq i \\ \perp & \text{otherwise} \end{cases}$
(IF ₁ -sym)	$\frac{\mathcal{V}(b)(E)}{\text{if } b \text{ then } s_1 \text{ else } s_2 \text{ fi}, E \rightarrow_{id} s_1, E}$
(IF ₂ -sym)	$\frac{\neg \mathcal{V}(b)(E)}{\text{if } b \text{ then } s_1 \text{ else } s_2 \text{ fi}, E \rightarrow_{id} s_2, E}$
(WHILE-sym)	$\frac{}{\text{while } b \text{ do } s \text{ od}, E \rightarrow_{id} \text{if } b \text{ then } s; \text{while } b \text{ do } s \text{ od else skip fi}, E}$
(SEQ ₁ -sym)	$\frac{s_1, E \rightarrow_\lambda s'_1, E'}{s_1; s_2, E \rightarrow_\lambda s'_1; s_2, E'}$
(SEQ ₂ -sym)	$\frac{}{\text{skip}; s_2, E \rightarrow_{id} s_2, E}$
(PAR ₁ -sym)	$\frac{1 \leq j \leq k \wedge s_j, E \rightarrow_\lambda s'_j, E'}{s_1 \parallel \dots \parallel s_j \parallel \dots \parallel s_k, E \rightarrow_\lambda s_1 \parallel \dots \parallel s'_j \parallel \dots \parallel s_k, E'}$
(PAR ₂ -sym)	$\frac{}{\text{skip} \parallel \dots \parallel \text{skip}, E \rightarrow_{id} \text{skip} \parallel \dots \parallel \text{skip}, E}$

- (DEL-sym) Straightforward.

(IF₁-sym), (IF₂-sym), (WHILE-sym), (SEQ₁-sym), (SEQ₂-sym), (PAR₁-sym) and (PAR₂-sym) are similar to the corresponding rules of the concrete semantics and are not explained further.

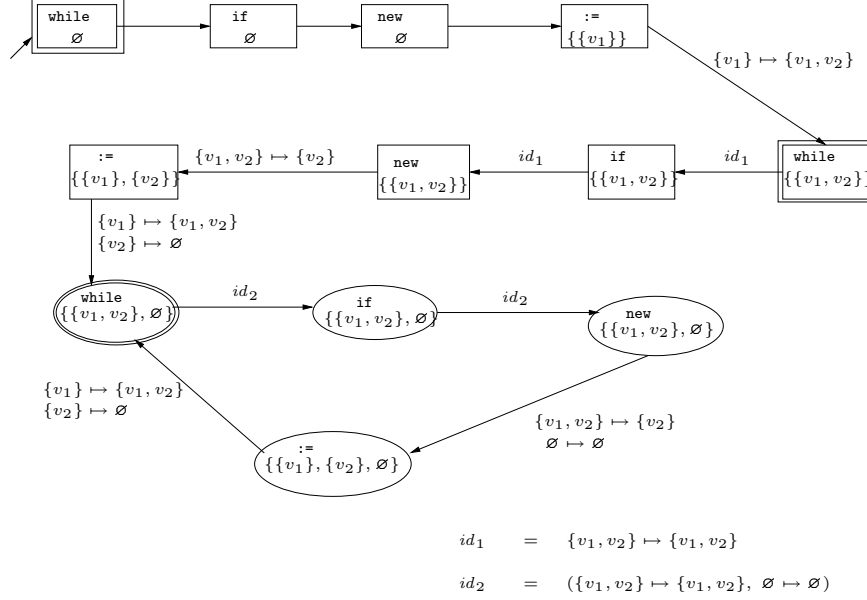


Figure 9: Symbolic semantics of $\text{decl } v_1, v_2 : (\text{while tt do new}(v_1); v_2 := v_1 \text{ od})$.

Example 4.4. Consider the program

$$p \equiv \text{decl } v_1, v_2 : \text{while tt do new}(v_1); v_2 := v_1 \text{ od}$$

of Example 4.3 and the corresponding abbreviations:

- **while** for “while tt do new(v_1); $v_2 := v_1$ od; skip”
- **if** for “if tt then new(v_1); $v_2 := v_1$; while else skip fi”
- **new** for “new(v_1); $v_2 := v_1$; while”
- **:=** for “ $v_2 := v_1$; while”.

Figure 9 depicts the symbolic semantics \mathcal{H}_p of p . Note how by using black-hole abstraction and reallocations, the automaton is *finite state* whereas \mathcal{A}_p is infinite.

4.4 Relating the concrete and symbolic semantics

The next theorem relates concrete and symbolic semantics.

Theorem 4.5. For any $p \in \mathcal{L}$: \mathcal{A}_p is an expansion of \mathcal{H}_p .

It thus follows that \mathcal{H}_p and \mathcal{A}_p are equivalent. Due to Corollary 3.12 they accept the same language up to isomorphism and therefore they satisfy the same ATL-formulae. Nevertheless, the symbolic semantics has the important property ensured by the following result.

Theorem 4.6. For any $p \in \mathcal{L}$: \mathcal{H}_p is finite state.

More precisely, for B_k the number of partitions of a set of k elements, $|s_{max}|$ the size of the longest sequential statement in p and m the number of sequential components in p , we have:

$$|Q_{\mathcal{H}_p}| \leq |s_{max}|^m \cdot \left[1 + 2 \cdot \sum_{k=1}^{|PVar|} \binom{|PVar|}{k} B_k \right].$$

B_k is known as the *Bell number* and has a non-trivial asymptotic behaviour [21], nevertheless, $O(2^n) < O(B_n) < O(n!)$. Thus, $Q_{\mathcal{H}}$ is exponential in the number of sequential components m in p . Whereas if m is constant then $|Q_{\mathcal{H}_p}|$ can be approximated by $O(2^{|PVar|} \cdot B_{|PVar|})$. As the symbolic semantics is deterministic, the number of transitions is of the order $O(m \cdot |Q_{\mathcal{H}_p}|)$.

Although the result given in this section holds for a simple language, we believe that it may be extended to more interesting programming languages. For instance, \mathcal{L} may be enhanced, in order to model the precise mechanism of creation and destruction of objects in an object-oriented programming languages. The interesting research question would then be if it is still possible to obtain finite state HABA.

5 Model-checking ATL

In this section, we define an algorithm for model-checking ATL formulae against a HABA. The algorithm extends the tableau method for LTL [17] to ATL.

We will evaluate ATL-formulae on states of a HABA by mapping the free variables of the formula to entities of the state. It should be clear that, in principle, any such mapping resolves all basic propositions: a freshness proposition x *new* holds if and only if x is mapped to an entity that is new in the state, and an entity equation $x = y$ holds if and only if x and y are mapped to the same entity. In turn, the basic propositions determine the validity of arbitrary formulae.

There are, however, two obstacles to this principle, the first of which is slight and the other more difficult to overcome.

- It is not always uniquely determined whether or not an entity is fresh in a state. Our model allows states in which a given entity is considered fresh when arriving by one incoming transition (since it is not in the codomain of the reallocation associated with that transition), but not when arriving by another (since according to the reallocation, the entity is the image of an entity in the previous state).

This obstacle is dealt with by *duplicating* the states where such an ambiguity exists. In general, there will therefore be as many duplicates of a given state as it has incoming transitions with distinct codomains.

- For variables (of the formula in question) that are mapped to the black hole, entity equations are not resolved, since it is not clear whether the variables are mapped to distinct entities that have imploded into the black hole, or to the same one.

To deal with this obstacle, we introduce an intermediate layer in the evaluation of the formula on the state. This additional layer consists of a *partial partitioning* of the free variables; that is, a set of nonempty, disjoint subsets of the set of all free variables. An entity equation is then resolved by the question whether the equated variables are in

the same partition. It is the partitions, rather than the individual variables, that are mapped to the entities of the state.

5.1 Duplication

The first aforementioned problem is illustrated by the HABA \mathcal{H} in Figure 10. Here, entity e_3 in state q_2 is old if transition $q_1 \rightarrow_{\lambda_{12}} q_2$ is taken. On the contrary, e_3 is new in q_2 if we consider the transition $q_3 \rightarrow_{\lambda_{32}} q_2$ (where $\lambda_{32} = \emptyset$). Furthermore, in the initial state q_3 the entity e_2 is new. However, it becomes old after the transition $q_4 \rightarrow_{\lambda_{43}} q_3$.

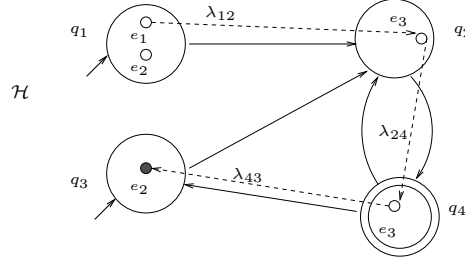


Figure 10: Ambiguity of the “new” entities.

Definition 5.1. For a HABA $\mathcal{H} = \langle X, Q, E, \rightarrow, I, \mathcal{F} \rangle$, the *duplication* of \mathcal{H} is the HABA $\mathcal{H}_\delta = \langle X, Q', E', \rightarrow', I', \mathcal{F}' \rangle$ where

- $Q' = \{(q, E_q \setminus \text{cod}(\lambda)) \mid q \in Q \wedge q' \rightarrow_\lambda q\} \cup \{(q, N) \mid I(q) = (N, \theta)\}$;
- $E'(q, M) = E_q$;
- \rightarrow' is defined by the following rule:

$$\frac{q \rightarrow_\lambda q'}{(q, M) \rightarrow'_\lambda (q', E_{q'} \setminus \text{cod}(\lambda))}$$

- $I' : (q, N) \mapsto \begin{cases} (N, \theta) & \text{if } I(q) = (N, \theta) \\ \text{undefined} & \text{otherwise} \end{cases}$
- $\mathcal{F}' = \{\{(q, M) \in Q' \mid q \in F_i\} \mid F_i \in \mathcal{F}\}$.

The set of variables X is unchanged. States are pairs (q, M) where $M \subseteq E_q$ is the subset of entities that are considered new in (q, M) . Initial states are defined according to initial valuations in I . In the definition of the transition relation, for every $q \rightarrow_\lambda q'$ in \mathcal{H} , a corresponding transition $(q, M) \rightarrow'_\lambda (q', M')$ in \mathcal{H}_δ is defined, provided that M' corresponds precisely to the set that are new according to λ , that is $M' = E_{q'} \setminus \text{cod}(\lambda)$. The set \mathcal{F}' contains those states resulting by the duplication of original accept states.

Example 5.2. The duplication \mathcal{H}_δ of the HABA in Figure 10 is shown in Figure 11. The original state q_2 is duplicated in (q_2, \emptyset) (where e_3 is old) and $(q_2, \{e_3\})$ (where e_3 is new). The initial state $(q_3, \{e_2\})$ is explicitly added in order to have e_2 new.

A HABA and its duplication are equivalent automata as stated by the following lemma.

Lemma 5.3. $\mathcal{L}(\mathcal{H}) = \mathcal{L}(\mathcal{H}_\delta)$.

Proof. See Appendix D. □

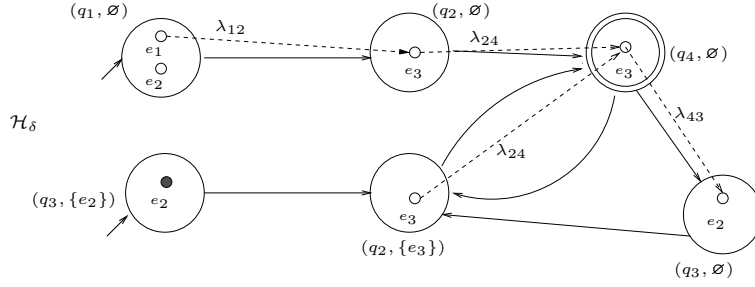


Figure 11: The duplication \mathcal{H}_δ of the HABA in Fig. 10.

Assumptions. In the remainder, we assume that the necessary duplication has been carried out already: that is, we will assume that a state $q \in \mathcal{Q}$ is a pair where the second component is a set $N_q \subseteq E_q$ that contains the entities that are *new in* q ; i.e., such that

- $q' \rightarrow_\lambda q$ implies $E_q \setminus \text{cod}(\lambda) = N_q$
- $I(q) = (N, \theta)$ implies $N = N_q$.

Note that, we can henceforth assume that I has just θ as its image — the component N is now uniquely associated with q .

Another assumption needed below is that every quantified variable actually appears free in the subformula; that is, we only consider formulae $\exists x.\phi$ for which $x \in \text{fv}(\phi)$. Note that this imposes no real restriction, since $\exists x.\phi$ is equivalent to $\exists x.(x \text{ alive} \wedge \phi)$.

Before we can present the model checking construction, we have to introduce a number of auxiliary notions.

5.2 Valuations

A valuation of a formula in a given state is an interpretation of the free variables of the formula as entities of the state. Such an interpretation establishes the validity of at least the *atomic propositions* within the formula, i.e., the sub-formulae of the form $x = y$ (which holds if x and y are interpreted as the same entity) and x *new* (which holds if x is interpreted as a fresh entity). Because we also want to allow the black hole as an interpretation, though, it is not enough to have a simple mapping from variables to entities: such a mapping still would not reveal whether two entities mapped to the black hole are mapped to *the same instance* imploded into the black hole. Therefore we first collect the variables into disjoint sets, the elements of which are considered equal, and map these sets of variables to entities.

Definition 5.4 (Valuations). Let $E \subseteq \text{Ent}^\infty$. An E -valuation is a triple (ϕ, Ξ, Θ) where ϕ is an ATL-formula and

- Ξ is a partial partitioning of $\text{fv}(\phi)$; that is, $\Xi = \{X_1, \dots, X_n\}$ such that $\emptyset \subset X_i \subseteq \text{fv}(\phi)$ for $1 \leq i \leq n$ and $X_i \cap X_j = \emptyset$ for $1 \leq i < j \leq n$ (but not necessarily $\bigcup_i X_i = \text{fv}(\phi)$, which would make it a *full* partitioning).
- $\Theta: \Xi \rightarrow E$ is a function mapping the partitions of Ξ to E , such that Θ is injective where it maps away from ∞ — i.e., $\Theta(X_i) = \Theta(X_j) \neq \infty$ implies $i = j$.

This is easily lifted to the states of a HABA: (ϕ, Ξ, Θ) is a q -valuation (for some $q \in Q_{\mathcal{H}}$) if it is an E_q -valuation (if $\lceil q \rceil$) or E_q^∞ -valuation (if $\lfloor q \rfloor$). We write $V_q(\phi)$, ranged over by v , to denote the set of q -valuations of ϕ , and V_q to denote the set of *all* q -valuations. We denote the components of a valuation v as $(\phi_v, \Xi_v, \Theta_v)$.

From such a partition interpretation Θ we can easily construct a “proper” (partial) interpretation $\bar{\Theta}: fv(\phi) \rightarrow Ent^\infty$ by *flattening* Θ :

$$\bar{\Theta}: x \mapsto \Theta(X) \quad \text{if } x \in X \in \text{dom}(\Theta).$$

A technicality: below we will need to restrict partial partitioning Ξ and mappings Θ of a valuation (ϕ, Ξ, Θ) to subformulae of ϕ , which means restricting the underlying sets of (free) variables upon which Ξ and Θ are built to those of that subformula. For this purpose, we define

$$\begin{aligned} \Xi \upharpoonright \psi &= \{X \cap fv(\psi) \mid X \in \Xi, X \cap fv(\psi) \neq \emptyset\} \\ \Theta \upharpoonright \psi &= \{(X \cap fv(\psi), \Theta(X)) \mid X \in \text{dom}(\Theta), X \cap fv(\psi) \neq \emptyset\}. \end{aligned}$$

The *atomic proposition valuations* of a state q of a HABA are those q -valuations of basic propositions of ATL (i.e., freshness predicates and entity equations) that make the corresponding properties true.

Definition 5.5. Let \mathcal{H} be a HABA and let $q \in Q_{\mathcal{H}}$ be arbitrary. The *atomic proposition valuations* of q are defined by the set $AV_q \subseteq V_q$ of all triples (ϕ, Ξ, Θ) for which one of the following holds:

- $\phi = \text{tt}$;
- $\phi = (x = y)$, and $x, y \in X$ for some $X \in \Xi$;
- $\phi = (x \text{ new})$, and $x \in X$ for some $X \in \Xi$ such that $\Theta(X) \in N_q$.

Closure. Along the lines of [17], we associate to each state q of a HABA a set of q -valuations, specifically aimed at establishing the validity of a given formula ϕ . For this purpose, we first collect all ATL-formulae whose validity is possibly relevant to the validity of ϕ into the so-called *closure* of ϕ . This includes especially all subformulae of ϕ , but also $\neg\psi$ if ψ is in the closure, $\mathbf{X}\neg\psi$ if $\mathbf{X}\psi$ is in the closure, and $\mathbf{X}(\psi_1 \cup \psi_2)$ if $\psi_1 \cup \psi_2$ is in the closure. Formally:

Definition 5.6. Let ϕ be an ATL-formula. The *closure* of ϕ , $CL(\phi)$, is the smallest set of formulae (identifying $\neg\neg\psi$ with ψ) such that:

- $\phi, \text{tt}, \text{ff} \in CL(\phi)$;
- $\neg\psi \in CL(\phi)$ iff $\psi \in CL(\phi)$;
- if $\psi_1 \vee \psi_2 \in CL(\phi)$ then $\psi_1, \psi_2 \in CL(\phi)$;
- if $\exists x.\psi \in CL(\phi)$ then $\psi \in CL(\phi)$;
- if $\mathbf{X}\psi \in CL(\phi)$ then $\psi \in CL(\phi)$;
- if $\neg\mathbf{X}\psi \in CL(\phi)$ then $\mathbf{X}\neg\psi \in CL(\phi)$;
- if $\psi_1 \cup \psi_2 \in CL(\phi)$ then $\psi_1, \psi_2, \mathbf{X}(\psi_1 \cup \psi_2) \in CL(\phi)$.

Example 5.7. The closure of the formula $\phi_1 \equiv \exists x.(x \text{ new} \wedge x \neq y)$ is:

$$CL(\phi_1) = \{\text{tt}, \text{ff}, \phi_1, \neg\phi_1, (x \text{ new} \wedge x \neq y), \neg(x \text{ new} \wedge x \neq y), x \text{ new}, \neg(x \text{ new}), x \neq y, x = y\}.$$

Similarly, the closure of $\phi_2 \equiv \forall x.x \neq y$ is

$$CL(\phi_2) = \{\text{tt}, \text{ff}, \phi_2, \neg\phi_2, \forall x.x \neq y, \neg\forall x.x \neq y, \exists x.x \neq y, \neg(\exists x.x \neq y), x \neq y, x = y\}.$$

Since valuations map (sets of) variables of a given formula to entities, possibly to the black hole, it is important to know how many of these variables have to be taken into account at the most. This is obviously bounded by the number of variables occurring (free or bound) in ϕ , but in fact we can be a little more precise: the number is given by $K(\phi)$ defined as

$$K(\phi) = \max \{|fv(\psi)| \mid \psi \in CL(\phi)\} .$$

The interesting case for the model checking construction is when one or more variables are indeed mapped to the black hole. Among other things, we will then have to make sure that sufficiently many entities of the state have imploded into the black hole to meet the demands of the valuation. For this purpose, we introduce the *black number* of a function, which is the number of entities that that function maps (implodes) into the black hole. For an arbitrary set A and (partial) mapping $\alpha: A \rightarrow Ent^\infty$ this is defined by

$$\Omega(\alpha) = |\{a \in A \mid \alpha(a) = \infty\}| .$$

5.3 Tableau graph

We now construct a graph that will be the basis of the model checking algorithm. The nodes of this graph, called *atoms* after [17], are built from states of a HABA, valuations of formulae from the closure, and a bound on the black number.

Definition 5.8. Given a HABA \mathcal{H} and an ATL-formula ϕ , an *atom* is a triple (q, D, k) where $q \in Q_{\mathcal{H}}$, $D \subseteq \{v \in V_q(\psi) \mid \psi \in CL(\phi), \Omega(\Theta_v) \leq k\}$ and $k \leq K(\phi)$ if $[q]$ or $k = 0$ if $[q]$, such that for all $v = (\psi, \Xi, \Theta) \in V_q$ with $\psi \in CL(\phi)$ and $\Omega(\Theta) \leq k$:

- if $v \in AV_q$, then $v \in D$;
- if $\psi = \neg\psi'$, then $v \in D$ iff $(\psi', \Xi, \Theta) \notin D$;
- if $\psi = \psi_1 \vee \psi_2$, then $v \in D$ iff $(\psi_i, \Xi \upharpoonright \psi_i, \Theta \upharpoonright \psi_i) \in D$ for $i = 1$ or $i = 2$;
- if $\psi = \exists x.\psi'$, then $v \in D$ iff there exists a $(\psi', \Xi', \Theta') \in D$ such that $\Xi = \Xi' \upharpoonright \psi$, $\Theta = \Theta' \upharpoonright \psi$ and $x \in \bigcup \Xi'$;
- if $\psi = \forall x.\psi'$, then $v \in D$ iff $(\psi', \Xi, \Theta) \in D$;
- if $\psi = \psi_1 \cup \psi_2$, then $v \in D$ iff either $(\psi_2, \Xi \upharpoonright \psi_2, \Theta \upharpoonright \psi_2) \in D$, or both $(\psi_1, \Xi \upharpoonright \psi_1, \Theta \upharpoonright \psi_1) \in D$ and $(\forall \psi, \Xi, \Theta) \in D$.

The set of all atoms for a given formula ϕ constructed on top of \mathcal{H} is denoted $A_{\mathcal{H}}(\phi)$, ranged over by A, B . We denote the components of an atom A by (q_A, D_A, k_A) .

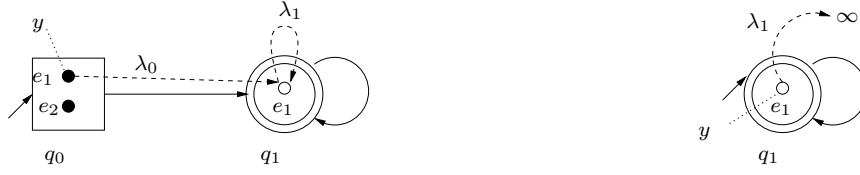


Figure 12: HABA \mathcal{H}_1 (left) and \mathcal{H}_2 (right).

Example 5.9. Consider the HABA \mathcal{H}_1 depicted in the left part of Figure 12 where e_1 and e_2 are new in q_0 and $X = \{y\}$. Although we will give an example for an until formula at the end of this section, for the moment recall the formula $\phi_1 \equiv \exists x.(x \text{ new} \wedge x \neq y)$ and its closure from Example 5.7. We compute the set of atoms $A_{\mathcal{H}_1}(\phi_1)$. Since $[q_0]$, for q_0 there is only one atom which is of the form $(q_0, D_0, 0)$. According to Def. 5.5 for the component D_0 the atomic proposition valuation $(\psi, \Xi, \Theta) \in AV_{q_0}$ such that $\psi \in CL(\phi_1)$ are the following:

$$v_0 = (\text{tt}, \emptyset, \emptyset)$$

$$v_1 = (x \text{ new}, \{\{x\}\}, \{x\} \mapsto e_1)$$

$$v_2 = (x \text{ new}, \{\{x\}\}, \{x\} \mapsto e_2)$$

$$v_3 = (x = y, \{\{x, y\}\}, \{x, y\} \mapsto e_1)$$

$$v_4 = (x = y, \{\{x, y\}\}, \{x, y\} \mapsto e_2).$$

Thus, by the first clause of Def. 5.8 we have $v_0, v_1, v_2, v_3, v_4 \in D_0$. By the clause for negation of Def. 5.8 we have:

$$v_5 = (x \neq y, \emptyset, \emptyset) \in D_0$$

$$v_6 = (x \neq y, \{\{x\}\}, \{x\} \mapsto e_1) \in D_0$$

$$v_7 = (x \neq y, \{\{x\}\}, \{x\} \mapsto e_2) \in D_0$$

$$v_8 = (x \neq y, \{\{y\}\}, \{y\} \mapsto e_1) \in D_0$$

$$v_9 = (x \neq y, \{\{y\}\}, \{y\} \mapsto e_2) \in D_0$$

$$v_{10} = (x \neq y, \{\{x\}, \{y\}\}, \{x\} \mapsto e_1, \{y\} \mapsto e_2) \in D_0$$

$$v_{11} = (x \neq y, \{\{x\}, \{y\}\}, \{x\} \mapsto e_2, \{y\} \mapsto e_1) \in D_0$$

$$v_{12} = (\neg(x \text{ new}), \emptyset, \emptyset) \in D_0.$$

Furthermore,

$$v_6, v_1 \in D_0 \Rightarrow v_{13} = (x \text{ new} \wedge x \neq y, \{\{x\}\}, \{x\} \mapsto e_1) \in D_0$$

$$v_7, v_2 \in D_0 \Rightarrow v_{14} = (x \text{ new} \wedge x \neq y, \{\{x\}\}, \{x\} \mapsto e_2) \in D_0$$

$$v_{10}, v_1 \in D_0 \Rightarrow v_{15} = (x \text{ new} \wedge x \neq y, \{\{x\}, \{y\}\}, \{x\} \mapsto e_1, \{y\} \mapsto e_2) \in D_0$$

$$v_{11}, v_2 \in D_0 \Rightarrow v_{16} = (x \text{ new} \wedge x \neq y, \{\{x\}, \{y\}\}, \{x\} \mapsto e_2, \{y\} \mapsto e_1) \in D_0.$$

By negation, we obtain

$$v_{17} = (\neg(x \text{ new} \wedge x \neq y), \emptyset, \emptyset) \in D_0$$

$$v_{18} = (\neg(x \text{ new} \wedge x \neq y), \{\{y\}\}, \{y\} \mapsto e_1) \in D_0$$

$$v_{19} = (\neg(x \text{ new} \wedge x \neq y), \{\{y\}\}, \{y\} \mapsto e_2) \in D_0$$

$$v_{20} = (\neg(x \text{ new} \wedge x \neq y), \{\{x, y\}\}, \{x, y\} \mapsto e_1) \in D_0$$

$$v_{21} = (\neg(x \text{ new} \wedge x \neq y), \{\{x, y\}\}, \{x, y\} \mapsto e_2) \in D_0.$$

By the clause on existential quantification:

$$\begin{aligned}
v_{13}, v_{14} \in D_0 &\Rightarrow v_{22} = (\exists x.(x \text{ new} \wedge x \neq y), \emptyset, \emptyset) \in D_0 \\
v_{15} \in D_0 &\Rightarrow v_{23} = (\exists x.(x \text{ new} \wedge x \neq y), \{\{y\}\}, \{y\} \mapsto e_2) \in D_0 \\
v_{16} \in D_0 &\Rightarrow v_{24} = (\exists x.(x \text{ new} \wedge x \neq y), \{\{y\}\}, \{y\} \mapsto e_1) \in D_0.
\end{aligned}$$

In D_0 there are no valuations for $\neg\exists x.(x \text{ new} \wedge x \neq y)$.

For state q_1 we have atoms $(q_1, D_1, 0)$, $(q_1, D_2, 1)$, $(q_1, D_3, 2)$. Sets D_1 , D_2 and D_3 differ because of the black number. In fact, valuation (ψ, Ξ, Θ) in D_1 must have $\Omega(\Theta) = 0$, while in D_2 and D_3 it must be $\Omega(\Theta) \leq 1$ and $\Omega(\Theta) \leq 2$, respectively. D_1 , D_2 , and D_3 are computed following the same pattern used for D_0 (and in fact many valuations $v \in D_0$ are also in D_1 , D_2 and D_3). In particular, D_1 contains the following valuations:

$$\begin{aligned}
v_0 &= (\mathbf{tt}, \emptyset, \emptyset) \\
v_3 &= (x = y, \{\{x, y\}\}, \{x, y\} \mapsto e_1) \\
v_5 &= (x \neq y, \emptyset, \emptyset) \\
v_6 &= (x \neq y, \{\{x\}\}, \{x\} \mapsto e_1) \\
v_8 &= (x \neq y, \{\{y\}\}, \{y\} \mapsto e_1).
\end{aligned}$$

Note that there are no valuations $(x \text{ new}, \Xi, \Theta)$ for any Ξ, Θ . Therefore, by negation, in D_1 we have

$$\begin{aligned}
v_{25} &= (\neg(x \text{ new}), \emptyset, \emptyset) \\
v_{26} &= (\neg(x \text{ new}), \{\{x\}\}, \{x\} \mapsto e_1) \\
v_{17} &= (\neg(x \text{ new} \wedge x \neq y), \emptyset, \emptyset) \\
v_{27} &= (\neg(x \text{ new} \wedge x \neq y), \{\{x\}\}, \{x\} \mapsto e_1) \\
v_{18} &= (\neg(x \text{ new} \wedge x \neq y), \{\{y\}\}, \{y\} \mapsto e_1) \\
v_{20} &= (\neg(x \text{ new} \wedge x \neq y), \{\{x, y\}\}, \{x, y\} \mapsto e_1).
\end{aligned}$$

Since there are no valuations for $x \text{ new} \wedge x \neq y$, it implies there are no valuations for $\exists x.x \text{ new} \wedge x \neq y$. This, in turn, implies by negation:

$$\begin{aligned}
v_{28} &= (\neg\exists x.(x \text{ new} \wedge x \neq y), \emptyset, \emptyset) \\
v_{29} &= (\neg\exists x.(x \text{ new} \wedge x \neq y), \{\{y\}\}, \{y\} \mapsto e_1).
\end{aligned}$$

The set $D_2 = D_1 \cup B_1$, where B_1 (valuation with black number 1) contains:

$$\begin{aligned}
v_{30} &= (x = y, \{\{x, y\}\}, \{x, y\} \mapsto \infty) \\
v_{31} &= (x \neq y, \{\{x\}\}, \{x\} \mapsto \infty) \\
v_{32} &= (x \neq y, \{\{y\}\}, \{y\} \mapsto \infty) \\
v_{33} &= (x \neq y, \{\{x\}, \{y\}\}, \{x\} \mapsto e_1, \{y\} \mapsto \infty) \\
v_{34} &= (x \neq y, \{\{x\}, \{y\}\}, \{x\} \mapsto \infty, \{y\} \mapsto e_1) \\
v_{35} &= (\neg(x \text{ new} \wedge x \neq y), \{\{x\}\}, \{x\} \mapsto \infty) \\
v_{36} &= (\neg(x \text{ new} \wedge x \neq y), \{\{y\}\}, \{y\} \mapsto \infty) \\
v_{37} &= (\neg(x \text{ new} \wedge x \neq y), \{\{x, y\}\}, \{x, y\} \mapsto \infty) \\
v_{38} &= (\neg(x \text{ new} \wedge x \neq y), \{\{x\}, \{y\}\}, \{x\} \mapsto e_1, \{y\} \mapsto \infty) \\
v_{39} &= (\neg(x \text{ new} \wedge x \neq y), \{\{x\}, \{y\}\}, \{x\} \mapsto \infty, \{y\} \mapsto e_1) \\
v_{40} &= (\neg\exists x.(x \text{ new} \wedge x \neq y), \{\{y\}\}, \{y\} \mapsto \infty).
\end{aligned}$$

Finally, $D_3 = D_2 \cup B_2$ where B_2 (valuation with black number 2) contains:

$$\begin{aligned}
v_{41} &= (x \neq y, \{\{x\}, \{y\}\}, \{x\} \mapsto \infty, \{y\} \mapsto \infty) \\
v_{42} &= (\neg(x \text{ new} \wedge x \neq y), \{\{x\}, \{y\}\}, \{x\} \mapsto \infty, \{y\} \mapsto \infty) \\
v_{43} &= (\neg(x \text{ new} \wedge x \neq y), \{\{x\}, \{y\}\}, \{x\} \mapsto \infty, \{y\} \mapsto \infty).
\end{aligned}$$

As for D_1 , both in D_2 and D_3 we do not have any valuation of the kind $(\exists x.(x \text{ new} \wedge x \neq y), \Xi, \Theta)$. In fact in q_1 , the formula ϕ_1 does not hold.

Example 5.10. Using valuations v of the previous example, we discuss a more interesting case. Consider the formula $\phi_2 \equiv \mathsf{X}\exists x.x \neq y$ and its closure from Example 5.7. The set of atoms is

$$\begin{aligned}
A_{\mathcal{H}_1}(\phi_2) &= \{ (q_0, D_4, 0), (q_0, D'_4, 0), \\
&\quad (q_1, D_5, 0), (q_1, D'_5, 0), (q_1, D_6, 1), (q_1, D'_6, 1), (q_1, D_7, 2), (q_1, D'_7, 2) \}.
\end{aligned}$$

We compute its elements in details. Instead of a single atom for q_0 , since ϕ_2 involves a next operator, we construct two atoms $(q_0, D_4, 0)$, $(q_0, D'_4, 0)$. In fact a formula of the kind $\mathsf{X}\psi$ does not imply anything concerning the validity of ψ in q_0 . This is also the reason why in the Definition 5.8 we do not have a special clause for the X operator. Thus we distinguish two possibilities, one in which $\mathsf{X}\psi$ holds (case D_4) and one where $\neg\mathsf{X}\psi$ holds (case D'_4). By the same argument as the previous example, we obtain

$$v_0, v_3, \dots, v_{11} \in D_4, D'_4.$$

Moreover,

$$\begin{aligned}
v_6, v_7 \in D_4, D'_4 &\Rightarrow (\exists x.x \neq y, \emptyset, \emptyset) \in D_4, D'_4 \\
v_{10} \in D_4, D'_4 &\Rightarrow (\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_2) \in D_4, D'_4 \\
v_{11} \in D_4, D'_4 &\Rightarrow (\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_1) \in D_4, D'_4.
\end{aligned}$$

This implies that in D_4 and D'_4 there are no triples of the form $(\neg\exists x.x \neq y, \Xi, \Theta)$ since all possible partitionings of $fv(\exists x.x \neq y)$ are in valuations with the formula $\exists x.x \neq y$. We define

$$\begin{aligned}
(\mathsf{X}\exists x.x \neq y, \emptyset, \emptyset) &\in D_4, \\
(\mathsf{X}\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_1) &\in D_4, \\
(\mathsf{X}\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_2) &\in D_4,
\end{aligned}$$

while

$$\begin{aligned} (\neg X \exists x. x \neq y, \emptyset, \emptyset) &\in D'_4 \\ (\neg X \exists x. x \neq y, \{\{y\}\}, \{y\} \mapsto e_1) &\in D'_4 \\ (\neg X \exists x. x \neq y, \{\{y\}\}, \{y\} \mapsto e_2) &\in D'_4 \end{aligned}$$

which in turn implies (by definition of atom)

$$\begin{aligned} (X \neg(\exists x. x \neq y), \emptyset, \emptyset) &\in D'_4 \\ (X \neg(\exists x. x \neq y), \{\{y\}\}, \{y\} \mapsto e_1) &\in D'_4 \\ (X \neg(\exists x. x \neq y), \{\{y\}\}, \{y\} \mapsto e_2) &\in D'_4. \end{aligned}$$

$$\begin{aligned} (\neg X \neg(\exists x. x \neq y), \emptyset, \emptyset) &\in D_4 \\ (\neg X \neg(\exists x. x \neq y), \{\{y\}\}, \{y\} \mapsto e_1) &\in D_4 \\ (\neg X \neg(\exists x. x \neq y), \{\{y\}\}, \{y\} \mapsto e_2) &\in D_4. \end{aligned}$$

The atoms of state q_1 are²: $(q_1, D_5, 0)$, $(q_1, D'_5, 0)$, $(q_1, D_6, 1)$, $(q_1, D'_6, 1)$, $(q_1, D_7, 2)$, $(q_1, D'_7, 2)$. The computation of D_5 and D'_5 is similar to D_4 and D'_4 , therefore we skip intermediate steps and we indicate only the resulting valuations:

$$\begin{aligned} D_5 = \{ & v_0, v_3, v_5, v_6, v_8, \\ & (\exists x. x \neq y, \quad \emptyset, \quad \emptyset) \\ & (\neg \exists x. x \neq y, \quad \{\{y\}\}, \{y\} \mapsto e_1), \\ & (X \exists x. x \neq y, \quad \emptyset, \quad \emptyset), \\ & (X \exists x. x \neq y, \quad \{\{y\}\}, \{y\} \mapsto e_1), \\ & (\neg X \neg(\exists x. x \neq y), \quad \emptyset, \quad \emptyset), \\ & (\neg X \neg(\exists x. x \neq y), \quad \{\{y\}\}, \{y\} \mapsto e_1) \}. \end{aligned}$$

and

$$\begin{aligned} D'_5 = \{ & v_0, v_3, v_5, v_6, v_8, \\ & (\exists x. x \neq y, \quad \emptyset, \quad \emptyset) \\ & (\neg \exists x. x \neq y, \quad \{\{y\}\}, \{y\} \mapsto e_1), \\ & (\neg X \exists x. x \neq y, \quad \emptyset, \quad \emptyset), \\ & (\neg X \exists x. x \neq y, \quad \{\{y\}\}, \{y\} \mapsto e_1), \\ & (X \neg(\exists x. x \neq y), \quad \emptyset, \quad \emptyset), \\ & (X \neg(\exists x. x \neq y), \quad \{\{y\}\}, \{y\} \mapsto e_1) \}. \end{aligned}$$

For D_6 and D'_6 we take $D_5 \setminus \{(\neg \exists x. x \neq y, \{\{y\}\}, \{y\} \mapsto e_1)\}$ and $D'_5 \setminus \{(\neg \exists x. x \neq y, \{\{y\}\}, \{y\} \mapsto e_1)\}$ respectively, and we extend them in order to include valuation with black number 1 (recall that atoms with D_6 and D'_6 have $\Omega(\Theta) = 1$).

$$v_{30}, v_{31}, v_{32}, v_{33}, v_{34} \in D_6, D'_6$$

and

$$\begin{aligned} v_{33} \in D_6, D'_6 &\Rightarrow (\exists x. x \neq y, \{\{y\}\}, \{y\} \mapsto \infty) \in D_6, D'_6 \\ v_{34} \in D_6, D'_6 &\Rightarrow (\exists x. x \neq y, \{\{y\}\}, \{y\} \mapsto e_1) \in D_6, D'_6 \end{aligned}$$

in particular, in D_6 and D'_6 there are no triples $(\neg(\exists x. x \neq y), \Xi, \Theta)$ for any Ξ , and Θ . This conforms with the intuition that the black number is 1, i.e., there exists one entity in the black hole distinct from e_1 . Thus $\neg \exists x. x \neq y$ cannot hold in q_1 .

²Again the primed version of a set D will be used for formulae of the kind $\neg X\psi$.

Again at this point we must distinguish between D_6 and D'_6 . We define:

$$\begin{aligned}
& (\exists x.x \neq y, \emptyset, \emptyset) \in D_6 \\
& (\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_1) \in D_6 \\
& (\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto \infty) \in D_6 \\
& (\neg \exists x.x \neq y, \emptyset, \emptyset) \in D_6 \\
& (\neg \exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_1) \in D_6 \\
& (\neg \exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto \infty) \in D_6.
\end{aligned}$$

and

$$\begin{aligned}
& (\exists x.x \neq y, \emptyset, \emptyset) \in D'_6 \\
& (\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_1) \in D'_6 \\
& (\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto \infty) \in D'_6 \\
& (\neg \exists x.x \neq y, \emptyset, \emptyset) \in D'_6 \\
& (\neg \exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_1) \in D'_6 \\
& (\neg \exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto \infty) \in D'_6.
\end{aligned}$$

Finally, it is easy to define D_7 and D'_7 :

$$D_7 = D_6 \cup \{v_{41}\} \text{ and } D'_7 = D'_6 \cup \{v_{41}\}.$$

Definition 5.11. The *tableau graph* for a HABA \mathcal{H} and an ATL-formula ϕ , denoted $G_{\mathcal{H}}(\phi)$, consists of vertices $A_{\mathcal{H}}(\phi)$ and edges $\rightarrow \subseteq A_{\mathcal{H}}(\phi) \times (Ent^\infty \rightarrow Ent^\infty) \times A_{\mathcal{H}}(\phi)$ determined by

$$\begin{aligned}
(q, D, k) \rightarrow_\lambda (q', D', k') \quad \text{iff} \quad & q \rightarrow_\lambda q', \\
& \forall X\psi \in CL(\phi): (X\psi, \Xi, \Theta) \in D \Leftrightarrow (\psi, \Xi, \lambda \circ \Theta) \in D', \\
& k' = \begin{cases} \min(K(\phi), k + \Omega(\lambda)) & \text{if } \lfloor q' \rfloor \\ 0 & \text{if } \lceil q' \rceil. \end{cases}
\end{aligned}$$

Note that if \mathcal{H} is finite-state, then $G_{\mathcal{H}}(\phi)$ can be effectively constructed: the set of atoms is finite for every given state.

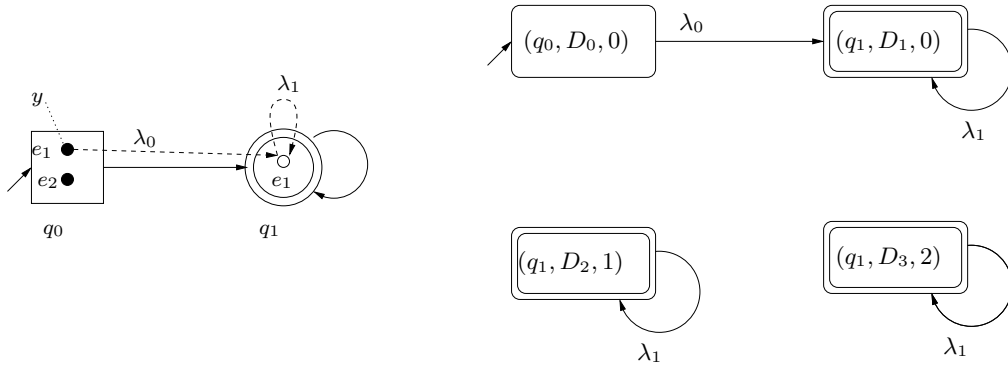


Figure 13: HABA \mathcal{H}_1 and corresponding graph $G_{\mathcal{H}_1}(\exists x.(x \text{ new} \wedge x \neq y))$.

Example 5.12. The graph $G_{\mathcal{H}_1}(\phi_1)$ for the set of atoms computed in Example 5.9 is shown in Figure 13 while $G_{\mathcal{H}_1}(\phi_2)$ for the set of atoms $A_{\mathcal{H}_1}(\phi_2)$ of Example 5.10 is shown in Figure 14. Atoms corresponding to accept states of the original HABA are drawn with a double circle³. In $G_{\mathcal{H}_1}(\phi_2)$, there are only four transitions (self-loops in atoms $(q_1, D_6, 1)$ and $(q_1, D_7, 2)$ and $(q_1, D_6, 1) \rightarrow_{\lambda_1} (q_1, D'_6, 1)$ and $(q_1, D_7, 2) \rightarrow_{\lambda_1} (q_1, D'_7, 2)$). No other transitions can be set because either the condition on the X operator or the condition on the black number of Def 5.11 is violated in any other case. For example, valuation $(\mathbf{X}\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_1)$ is in D_4 , however, $(\exists x.x \neq y, \{\{y\}\}, \lambda_0 \circ \{y\} \mapsto e_1) = (\exists x.x \neq y, \{\{y\}\}, \{y\} \mapsto e_1)$ does not belong to D_5 , therefore there is no transition $(q_0, D_4, 0) \rightarrow_{\lambda_0} (q_1, D_5, 0)$. While, $(q_0, D_4, 0) \not\rightarrow_{\lambda_0} (q_1, D_6, 1)$ because the condition on the k component is not fulfilled.

Example 5.13. The right part of Figure 12 shows the HABA \mathcal{H}_2 with the only state q_1 (it represents a modified version of q_1 of \mathcal{H}_1). In this case λ_1 implodes e_1 and therefore in each state a new entity is created. For $\phi_2 \equiv \mathbf{X}\exists x.x \neq y$ the set of atoms contains precisely those atoms obtained in the previous example for state q_1 . In fact, in the definition of atoms, the transitions of the HABA are not taken into account. Atoms for a given q only depend on E_q , N_q and on the (un)boundness of q . For \mathcal{H}_2 the resulting graph $G_{\mathcal{H}_2}(\phi_2)$ is depicted in the right part of Figure 15. In this case the third component k of the atom plays an active role. Since λ_1 implodes at each step an entity, the possible transitions in the graph are between atoms of the form $(\psi, D, k) \rightarrow_{\lambda_1} (\psi, D, k + 1)$ with $k = 0, 1$. This is because in general with a formula ϕ , we may need to distinguish up to $K(\phi)$ different situations according to how many entities are imploded into the black hole. Above this number we do not longer need to differentiate. This explains the transition $(q_1, D_7, 2) \rightarrow_{\lambda_1} (q_1, D_7, 2)$.

A *path* through a tableau graph is an infinite sequence of states and transitions, starting at an initial state of the HABA and satisfying the acceptance condition of the HABA, such that all “until”-subformulae in any of the atoms are satisfied somewhere further down the sequence.

Definition 5.14. An *allocational path* in $G_{\mathcal{H}}(\phi)$ is an infinite sequence $\pi = (q_0, D_0, k_0) \lambda_0 (q_1, D_1, k_1) \lambda_1 \cdots$ such that:

1. $q_0 \lambda_0 q_1 \lambda_1 \cdots \in \text{runs}(\mathcal{H})$;
2. for all $i \geq 0$, $(q_i, D_i, k_i) \rightarrow_{\lambda_i} (q_{i+1}, D_{i+1}, k_{i+1})$;
3. for all $i \geq 0$ and all $(\psi_1 \cup \psi_2, \Xi, \Theta) \in D_i$, there exists a $j \geq i$ such that $(\psi_2, \Xi \upharpoonright \psi_2, \lambda_{j-1} \circ \cdots \lambda_i \circ (\Theta \upharpoonright \psi_2)) \in D_j$.

Given an allocational path π in $G_{\mathcal{H}}(\phi)$ of this form, we say that π *fulfills* ϕ if the underlying run $\rho = q_0 \lambda_0 q_1 \lambda_1 \cdots$ generates an allocation triple (σ, N, θ) with a generator $(h_i)_{i \in \mathbb{N}}$ such that $k_0 = \min(K(\phi), \Omega(h_0))$ and $\sigma, N, \theta \models \phi$. If ϕ is clear from the context, we call π a *fulfilling path*. Furthermore, recall that (cf. Definition 3.11) if there exists $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{H})$ such that $\sigma, N, \theta \models \phi$ we say that ϕ is \mathcal{H} -satisfiable.

This sets the stage for the main results. We first state the correspondence between the fulfilment of a formula by a path and the presence of that formula in the initial atom of the path.

³Formally speaking a tableau graph does not have accept states. However, here we distinguish these atoms from the others only in order to facilitate their identification that will be useful later.

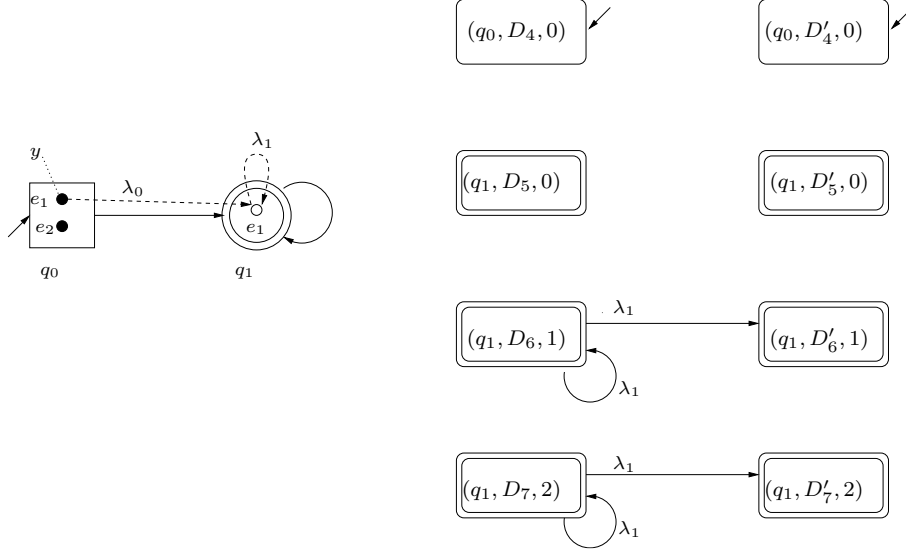


Figure 14: HABA \mathcal{H}_1 and corresponding graph $G_{\mathcal{H}_1}(\mathbb{X}(\exists x.x \neq y))$.

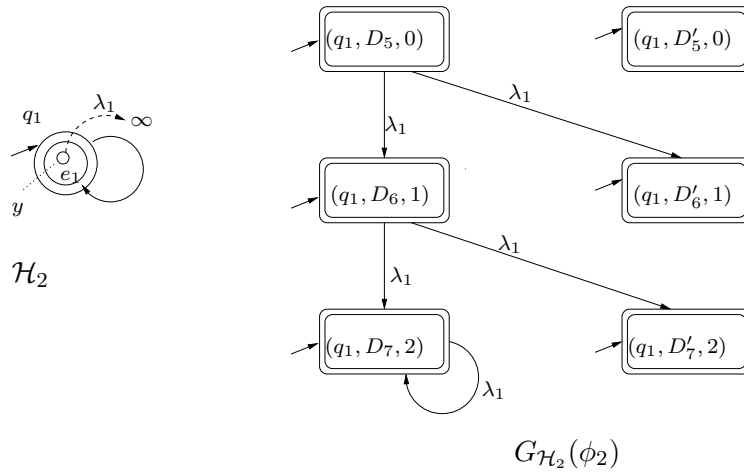


Figure 15: HABA \mathcal{H}_2 and corresponding graph $G_{\mathcal{H}_2}(\mathbb{X}(\exists x.x \neq y))$.

Proposition 5.15. A path π in $G_{\mathcal{H}}(\phi)$ fulfills ϕ if and only if there exists $(\phi, \Xi, \Theta) \in D_0$ (for some Ξ, Θ) such that $I_{\mathcal{H}}(q_0) = \overline{\Theta}$.

Proof. See Appendix D □

Furthermore, there is a correspondence between the satisfiability of a formula in the HABA and the existence of a fulfilling path in the tableau graph.

Proposition 5.16. ϕ is \mathcal{H} -satisfiable if and only if there exists a path in $G_{\mathcal{H}}(\phi)$ that fulfills ϕ .

Proof. See Appendix D □

Example 5.17. The path $\pi = (q_0, D_0, 0)\lambda_0((q_1, D_1, 0)\lambda_1)^\omega$ of $G_{\mathcal{H}}(\phi_1)$ in Figure 13 is fulfilling for $\phi_1 \equiv \exists x.(x \text{ new} \wedge x \neq y)$. On the other hand, ϕ_2 is not \mathcal{H}_1 -satisfiable. In fact, there are no paths in $G_{\mathcal{H}_1}(\phi_2)$. Finally, ϕ_2 is \mathcal{H}_2 -satisfiable. The path $(q_1, D_5, 0)\lambda_1(q_1, D_6, 1)(\lambda_1(q_1, D_7, 2))^\omega$ is a fulfilling path.

From now on we can (almost) rely on standard theory (see [17]). The first observation is that a tableau graph can have infinitely many different paths, therefore looking for a fulfilling path for ϕ is still not an effective method for model checking. We need the following definitions.

A subgraph $G' \subseteq G_{\mathcal{H}}(\phi)$ is *self-fulfilling* if every node A in G' has at least an outgoing edge and for every $(\psi_1 \cup \psi_2, \Xi, \Theta) \in D_A$ there exists a node $B \in G'$ such that

- $A = A_0 \rightarrow_{\lambda_0} A_1 \rightarrow_{\lambda_1} \cdots \rightarrow_{\lambda_{i-2}} A_{i-1} \rightarrow_{\lambda_{i-1}} A_i = B$
- $(\psi_2, \Xi \upharpoonright \psi_2, \lambda_{i-1} \circ \cdots \circ \lambda_0 \circ (\Theta \upharpoonright \psi_2)) \in D_B$.

A *prefix* in $G_{\mathcal{H}}(\phi)$ is a sequence $A_0 \rightarrow_{\lambda_0} A_1 \rightarrow_{\lambda_1} \cdots \rightarrow_{\lambda_{i-2}} A_{i-1} \rightarrow_{\lambda_{i-1}} A_i$ such that A_0 is an initial atom (i.e., $q_{A_0} \in I_{\mathcal{H}}$) and A_i is in a self-fulfilling subgraph.

Let $Inf(\pi)$ denote the set of nodes that appear infinitely often in the path π . $Inf(\pi)$ is a strongly connected subgraph (SCS). We can prove the following implications:

Proposition 5.18. π is a fulfilling path in $G_{\mathcal{H}}(\phi) \Rightarrow Inf(\pi)$ is a self-fulfilling SCS of $G_{\mathcal{H}}(\phi)$.

Proof. See Appendix D □

Proposition 5.19. Let $G' \subseteq G_{\mathcal{H}}(\phi)$ be a self-fulfilling SCS such that

- there exists a prefix of G' starting at an initial atom A with $(\phi, \Xi, \Theta) \in D_A$ such that $I_{\mathcal{H}}(q_A) = \overline{\Theta}$;
- for all $F \in \mathcal{F}_{\mathcal{H}} : F \cap \{q \mid (q, D, k) \in G'\} \neq \emptyset$;

Then there exists a path π in $G_{\mathcal{H}}(\phi)$ that fulfills ϕ such that $Inf(\pi) = G'$.

Proof. See Appendix D □

Finally, we can collect all the previous results into the following theorem, which is the main result of the paper, stating in essence that the model-checking problem for ATL over HABA is decidable.

Theorem 5.20. For any HABA \mathcal{H} and formula ϕ , it is decidable whether or not ϕ is \mathcal{H} -satisfiable.

Proof. See Appendix D □

Example 5.21. At this point, it becomes interesting to have an final example that summarises the complete model-checking procedure. Consider $\phi_3 \equiv \mathbf{G}(\exists x.x \neq y)$. This formula expresses that there are always at least two entities. It can be rewritten as $\phi_3 \equiv \neg(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y)$. Its closure is

$$\begin{aligned} CL(\phi_3) = \{ & \mathbf{tt}, \mathbf{ff}, \phi_3, \neg \phi_3, \exists x.x \neq y, \neg \exists x.x \neq y \\ & \mathbf{X}(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \neg \mathbf{X}(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \\ & \mathbf{X}\neg(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \neg \mathbf{X}\neg(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), x \neq y, x = y\}. \end{aligned}$$

We check whether ϕ_3 is \mathcal{H}_2 -satisfiable (\mathcal{H}_2 is depicted in Figure 12). Similarly to Example 5.10, the set of atoms is $A_{\mathcal{H}_2}(\phi_3) = \{(q_1, D_8, 0), (q_1, D'_8, 0), (q_1, D_9, 1), (q_1, D'_9, 1), (q_1, D_{10}, 2), (q_1, D'_{10}, 2)\}$. We just give the final resulting sets of valuation since the computation follows the same pattern as the previous examples⁴.

$$\begin{aligned} D_8 = \{ & v_0, v_3, v_5, v_6, v_8, \\ & (\exists x.x \neq y, \quad \emptyset), \\ & (\neg(\exists x.x \neq y), \quad \{y\} \mapsto e_1), \\ & (\mathbf{X}(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \quad \emptyset), \\ & (\mathbf{X}(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y)), \quad \{y\} \mapsto e_1), \\ & (\neg \mathbf{X}\neg(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \quad \emptyset), \\ & (\neg \mathbf{X}\neg(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y)), \quad \{y\} \mapsto e_1), \\ & ((\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \quad \emptyset), \\ & ((\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \quad \{y\} \mapsto e_1) \} \end{aligned}$$

$$\begin{aligned} D'_8 = \{ & v_0, v_3, v_5, v_6, v_8 \\ & (\exists x.x \neq y, \quad \emptyset), \\ & (\neg(\exists x.x \neq y), \quad \{y\} \mapsto e_1), \\ & (\neg \mathbf{X}(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \quad \emptyset), \\ & (\neg \mathbf{X}(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y)), \quad \{y\} \mapsto e_1), \\ & (\mathbf{X}\neg(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \quad \emptyset), \\ & (\mathbf{X}\neg(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y)), \quad \{y\} \mapsto e_1), \\ & (\neg(\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \quad \emptyset), \\ & ((\mathbf{tt} \mathbf{U} \neg \exists x.x \neq y), \quad \{y\} \mapsto e_1) \} \end{aligned}$$

⁴Here, for the sake of brevity, in a valuation (ψ, Ξ, Θ) we skip the second component Θ since it corresponds to $\text{dom}(\Theta)$.

Sets D_9 and D'_9 contain also valuations with black number 1.

$$\begin{aligned}
D_9 = & (D_8 \setminus \{(\neg(\exists x.x \neq y), \{y\} \mapsto e_1)\}) \cup \\
& \{ v_{30}, v_{31}, v_{32}, v_{33}, v_{34} \\
& (\exists x.x \neq y, \quad \emptyset), \\
& ((\exists x.x \neq y), \quad \{y\} \mapsto e_1), \\
& ((\exists x.x \neq y), \quad \{y\} \mapsto \infty), \\
& (\mathsf{X}(\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \emptyset), \\
& (\mathsf{X}(\mathsf{tt} \cup \neg \exists x.x \neq y)), \quad \{y\} \mapsto e_1), \\
& (\mathsf{X}(\mathsf{tt} \cup \neg \exists x.x \neq y)), \quad \{y\} \mapsto \infty), \\
& (\neg \mathsf{X} \neg (\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \emptyset), \\
& (\neg \mathsf{X} \neg (\mathsf{tt} \cup \neg \exists x.x \neq y)), \quad \{y\} \mapsto e_1), \\
& (\neg \mathsf{X} \neg (\mathsf{tt} \cup \neg \exists x.x \neq y)), \quad \{y\} \mapsto \infty), \\
& ((\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \emptyset), \\
& ((\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \{y\} \mapsto e_1), \\
& ((\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \{y\} \mapsto \infty) \}
\end{aligned}$$

$$\begin{aligned}
D'_9 = & (D'_8 \setminus \{(\neg(\exists x.x \neq y), \{y\} \mapsto e_1)\}) \cup \\
& \{ v_{30}, v_{31}, v_{32}, v_{33}, v_{34} \\
& (\exists x.x \neq y, \quad \emptyset), \\
& ((\exists x.x \neq y), \quad \{y\} \mapsto e_1), \\
& ((\exists x.x \neq y), \quad \{y\} \mapsto \infty), \\
& (\neg \mathsf{X}(\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \emptyset), \\
& (\neg \mathsf{X}(\mathsf{tt} \cup \neg \exists x.x \neq y)), \quad \{y\} \mapsto e_1), \\
& (\neg \mathsf{X}(\mathsf{tt} \cup \neg \exists x.x \neq y)), \quad \{y\} \mapsto \infty), \\
& (\mathsf{X} \neg (\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \emptyset), \\
& (\mathsf{X} \neg (\mathsf{tt} \cup \neg \exists x.x \neq y)), \quad \{y\} \mapsto e_1), \\
& (\mathsf{X} \neg (\mathsf{tt} \cup \neg \exists x.x \neq y)), \quad \{y\} \mapsto \infty), \\
& (\neg (\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \emptyset), \\
& (\neg (\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \{y\} \mapsto e_1), \\
& (\neg (\mathsf{tt} \cup \neg \exists x.x \neq y), \quad \{y\} \mapsto \infty) \}.
\end{aligned}$$

In particular note that in D_9 and D'_9 there are no triples of the form $(\neg \exists x.x \neq y, \Theta)$ for any Θ . Finally, we have $D_{10} = D_9 \cup \{v_{41}\}$ and $D'_{10} = D'_9 \cup \{v_{41}\}$.

The graph $G_{\mathcal{H}_2}(\phi_3)$ is depicted in Figure 16. The valuation $(\neg(\mathsf{tt} \cup \neg \exists x.x \neq y), \{y\} \mapsto e_1)$ belongs neither to D_8 nor D'_8 , therefore the formula $\phi_3 \equiv \mathsf{G}(\exists x.x \neq y)$ is not \mathcal{H}_2 -satisfiable. This is in fact correct since in the first state there is only one entity. However, it is interesting to note that $(\mathsf{X} \neg (\mathsf{tt} \cup \neg \exists x.x \neq y), \{y\} \mapsto e_1) \in D'_8$, therefore the path $\pi = (q_1, D'_8, 0) \lambda_1 (q_1, D'_9, 1) (\lambda_0 (q_1, D'_{10}, 2))^\omega$ is fulfilling for the the formula $\mathsf{X}\phi_3 \equiv \mathsf{XG}(\exists x.x \neq y)$. Hence, $\mathsf{X}\phi_3$ is \mathcal{H}_2 -satisfiable. Again this is correct since in the second state there is already an imploded entity in the black hole. Although for this example it easy to see that π fulfils $\mathsf{X}\phi_3$, Proposition 5.19 can also be used. In particular, $(q_1, D'_8, 0)$ is the initial atom containing the correct valuation of $\mathsf{X}\phi_3$. Furthermore, $(q_1, D'_8, 0) \lambda_1 (q_1, D'_9, 1)$ is a prefix and $(q_1, D'_{10}, 2) \lambda_1 (q_1, D'_{10}, 2)$ is a self-fulfilling SCS (note that in D'_{10} there are no valuations for U -formulae but only valuations for negations of U -formulae). Finally, the intersection between the SCS and the set $\mathcal{F}_{\mathcal{H}_2}$ is not empty since q_1 is the only accept state of \mathcal{H}_2 .

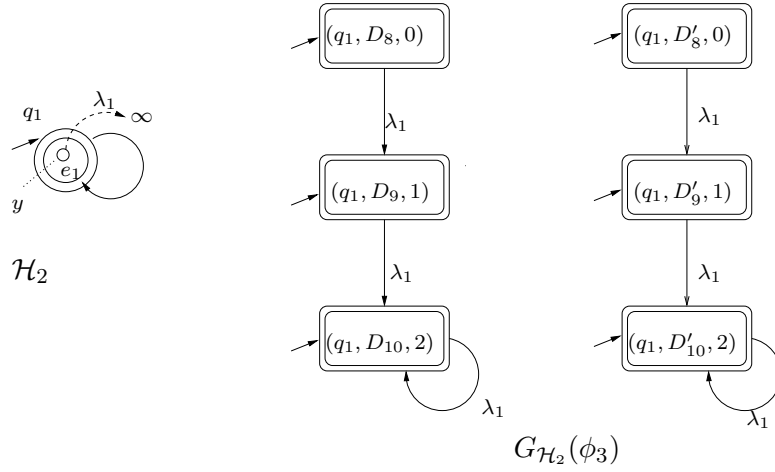


Figure 16: HABA \mathcal{H}_2 and corresponding graph $G_{\mathcal{H}_2}(\text{G}\exists x.x \neq y)$.

5.4 Complexity

We discuss briefly and in a rather informal manner on the complexity of the algorithm described in this section. The aim is to find an upper bound to the worst-case time complexity of model-checking. First we investigate the construction of the tableau graph. This involves as a preliminary step the duplication of the HABA. Then we give an upper bound on the number of steps needed to decide if a SCS is self-fulfilling.

Duplication For a HABA \mathcal{H} , let $T_{\max} = \max \{ |\{(q, \lambda, q') \mid (q, \lambda, q') \in \rightarrow_{\mathcal{H}}\}| \mid q, q' \in Q_{\mathcal{H}} \}$, i.e., T_{\max} is the largest number of transitions between two states of \mathcal{H} . The number of states of the duplication \mathcal{H}_{δ} is bounded by:

$$|Q_{\mathcal{H}_{\delta}}| \leq |Q_{\mathcal{H}}|^2 \cdot T_{\max} \quad (1)$$

as in the worst case for a given state $q \in Q_{\mathcal{H}}$ we have to create a new state $q' \in Q_{\mathcal{H}_{\delta}}$ for every incoming transitions of q .

The number of transitions we have to add to construct \mathcal{H}_{δ} is linear to $|Q_{\mathcal{H}_{\delta}}|$ because one new transition is enough for every new state.

Note that the largest number of transitions between two states does not increase after the duplication. Therefore when needed we will use T_{\max} of \mathcal{H} .

Graph construction. For a given ATL-formula ϕ , note that $O(|CL(\phi)|) = O(|\phi|)$ as well as $O(K(\phi)) = O(|\phi|)$. The construction of the graph is done on the duplication of \mathcal{H} . We have:

$$|A_{\mathcal{H}}(\phi)| \leq 2^{|\phi|} \cdot |Q_{\mathcal{H}_{\delta}}| \cdot |\phi| \quad (2)$$

because, for every state $q \in Q_{\mathcal{H}_{\delta}}$ we have $K(\phi)$ atoms if $\lfloor q \rfloor$ and each atom must be duplicated for every formula of the type $X\psi \in CL(\phi)$.

The complexity of the construction of a single atom (q, D, k) is dictated by the size of the set of valuations D . It is bounded by the following:

$$|D| \leq |\phi| \cdot B_{|\phi|} \cdot \frac{(|E_q| + 1)!}{(|E_q| + 1 - |\phi|)!} \quad (3)$$

where $B_{|\phi|}$ is the $|\phi|$ -th Bell number. This formula can be explained as follows: for a $\psi \in CL(\phi)$ we have a valuation for every partial partition of $fv(\psi)$. There are $B_{fv(\psi)}$ of these valuations. We can substitute $B_{fv(\psi)}$ by $B_{|\phi|}$ because $B_{|fv(\psi)|} \leq B_{|\phi|}$. For each of these partitions we need to consider every injection to E_q (last term of (3)). We add 1 in order to consider the black hole. Finally, since in D there are valuations for each $\psi \in CL(\phi)$, we multiply by the first term $|\phi|$.

Let $E_{\max} = \max \{|E_q| \mid q \in Q_{\mathcal{H}}\}$, i.e., E_{\max} is the cardinality of the largest set of entities in any state of \mathcal{H} (and therefore of \mathcal{H}_δ). Furthermore, let us denote with D_{\max} the cardinality of the largest set of valuations in $A_{\mathcal{H}}(\phi)$, i.e., $D_{\max} = \max \{|D| \mid (q, D, k) \in A_{\mathcal{H}}(\phi)\}$. Since $\frac{(|E_q|+1)!}{(|E_q|+1-|\phi|)!} \leq (|E_q|+1)^{|\phi|}$, then by (3) we obtain:

$$D_{\max} \leq |\phi| \cdot B_{|\phi|} \cdot (E_{\max} + 1)^{|\phi|}. \quad (4)$$

Thus, we conclude that the complexity for the construction of a single atom is of the order $O(|\phi| \cdot B_{|\phi|} \cdot E_{\max}^{|\phi|})$. The number of edges T_G in $G_{\mathcal{H}}(\phi)$ is bounded by⁵

$$T_G \leq |A_{\mathcal{H}}(\phi)|^2 \cdot T_{\max}. \quad (5)$$

Finally, the cost of the construction of $G_{\mathcal{H}}(\phi)$ is given by summing the complexity of building $A_{\mathcal{H}}(\phi)$ and the set of edges respectively. Using (1), (2), (4) and (5) we have:

$$\begin{aligned} & O(|A_{\mathcal{H}}(\phi)| \cdot D_{\max} + |A_{\mathcal{H}}(\phi)|^2 \cdot T_{\max}) \\ &= O(|A_{\mathcal{H}}(\phi)| \cdot |\phi| \cdot B_{|\phi|} \cdot E_{\max}^{|\phi|} + |A_{\mathcal{H}}(\phi)|^2 \cdot T_{\max}) \\ &= O(2^{|\phi|} \cdot |Q_{\mathcal{H}_\delta}| \cdot |\phi|^2 \cdot (B_{|\phi|} \cdot E_{\max}^{|\phi|} + 2^{|\phi|} \cdot |Q_{\mathcal{H}_\delta}| \cdot T_{\max})) \\ &= O(2^{|\phi|} \cdot |Q_{\mathcal{H}}|^2 \cdot T_{\max} \cdot |\phi|^2 \cdot (B_{|\phi|} \cdot E_{\max}^{|\phi|} + 2^{|\phi|} \cdot |Q_{\mathcal{H}}|^2 \cdot T_{\max}^2)). \end{aligned}$$

Thus, constructing the graph is:

- $O(|Q_{\mathcal{H}}|^4)$, i.e., polynomial in the number of states of \mathcal{H} ;
- $O(2^{|\phi|} \cdot |\phi|^2 \cdot B_{|\phi|} \cdot E_{\max}^{|\phi|})$, i.e., super-exponential on the size of ϕ ;
- $O(E_{\max}^{|\phi|})$, i.e., polynomial in the largest number of entities in \mathcal{H} .

Deciding if a SCS is self-fulfilling. To decide whether a given strongly connected component G is self-fulfilling, we propose an algorithm consisting of the following steps:

- Construct a linking structure with nodes $(A, \psi_1 \cup \psi_2, \Theta)$ where $A = (q, D, k)$ is an atom in G and $(\psi_1 \cup \psi_2, \Xi, \Theta) \in D$, and edges $(A', \psi_1 \cup \psi_2, \Theta') \rightarrow (A, \psi_1 \cup \psi_2, \Theta)$ for all $A \rightarrow_\lambda A'$ with $\Theta' = \lambda \circ \Theta$. (Note that the links in this newly created structure go in the reverse direction w.r.t. the edges of the tableau graph.)

⁵In principle, T_{\max} is of the order $O(E_{\max}^{E_{\max}})$ as we can have every possible injective partial mapping from a set of cardinality E_{\max} to another set of cardinality E_{\max} . However, it is reasonable to consider that for big E_{\max} such a bound becomes unrealistic. It is hard to imagine such a HABA with so many transitions between two states. For example, the symbolic semantics presented in Section 4 is deterministic. T_{\max} is equal to the number of parallel components of the program and does not depend on E_{\max} . By this consideration, for a more sensible general picture of the model checking algorithm computational cost, it seems reasonable to keep T_{\max} itself in the complexity measure.

- Appoint in this structure all nodes $(A, \psi_1 \cup \psi_2, \Theta)$ as *initial* for which $(\psi_2, \Xi \upharpoonright \psi_2, \Theta \upharpoonright \psi_2) \in D_A$.
- Perform a reachability analysis on the structure thus obtained.

We claim that G is self-fulfilling iff all nodes of this linking structure are reachable. For if all nodes are reachable then for all $A \in G$ and all $(\psi_1 \cup \psi_2, \Xi, \Theta) \in D_A$ there is a path

$$(A_n, \psi_1 \cup \psi_2, \Theta_n) \rightarrow \dots \rightarrow (A_0, \psi_1 \cup \psi_2, \Theta_0)$$

with $A_0 = A$, $\Theta_0 = \Theta$, $A_0 \rightarrow_{\lambda_0} \dots \rightarrow_{\lambda_{n-1}} A_n$ and $(\psi_2, \Xi \upharpoonright \psi_2, \lambda_{n-1} \circ \dots \circ \lambda_0 \circ (\Theta_0 \upharpoonright \psi_2)) \in D_{A_0}$; hence G is self-fulfilling. Vice versa, if for all $A \in G$ and all $(\psi_1 \cup \psi_2, \Xi, \Theta) \in D_A$ there is a node $B \in G$ such that

- $A = A_0 \rightarrow_{\lambda_0} \dots \rightarrow_{\lambda_{n-1}} A_n = B$
- $(\psi_2, \Xi \upharpoonright \psi_2, \lambda_{n-1} \circ \dots \circ \lambda_0 \circ (\Theta \upharpoonright \psi_2)) \in D_B$

then also $(A_n, \psi_1 \cup \psi_2, \Theta_n) \rightarrow \dots \rightarrow (A_0, \psi_1 \cup \psi_2, \Theta_0)$ with $\Theta_i = \lambda_{i-1} \circ \dots \circ \lambda_0 \circ \Theta$ is a path through the linking structure, and $(A_n, \psi_1 \cup \psi_2, \Theta_n)$ is an initial node in that structure.

The cost of this analysis equals the cost of building the linking structure plus the cost of the reachability analysis, which is linear in the size of that linking structure. The size of this structure is dictated by the number of edges, which equals the number of edges in G times the number of “U-valuations” in each atom of G — which in turn is linear to D_{\max} . Thus we obtain the following worst case cost for establishing whether a given SCS G is self-fulfilling:

$$O(T_G \cdot D_{\max})$$

where T_G is the number of edges in G .

The number of SCS in $G_{\mathcal{H}}(\phi)$ is of the order $O(2^{|A_{\mathcal{H}}(\phi)} \cdot T_{\max})$. Hence, the worst time complexity for checking if there exists a self-fulfilling SCS is:

$$\begin{aligned} & O(2^{|A_{\mathcal{H}}(\phi)} \cdot T_{\max} \cdot T_G \cdot D_{\max}) \\ &= O(2^{|Q_{\mathcal{H}_\delta} \cdot |\phi| \cdot 2^{|\phi|}} \cdot T_{\max}^2 \cdot 2^{2|\phi|} \cdot |Q_{\mathcal{H}_\delta}|^2 \cdot |\phi|^3 \cdot B_{|\phi|} \cdot E_{\max}^{|\phi|}) \\ &= O(2^{(2^{|\phi|} \cdot |Q_{\mathcal{H}_\delta}| + 2)|\phi|} \cdot T_{\max}^2 \cdot |Q_{\mathcal{H}_\delta}|^2 \cdot |\phi|^3 \cdot B_{|\phi|} \cdot E_{\max}^{|\phi|}) \\ &= O(2^{(2^{|\phi|} \cdot |Q_{\mathcal{H}}|^2 \cdot T_{\max} + 2)|\phi|} \cdot T_{\max}^4 \cdot |Q_{\mathcal{H}}|^4 \cdot |\phi|^3 \cdot B_{|\phi|} \cdot E_{\max}^{|\phi|}) \end{aligned}$$

If we single out the different parameters we have:

- $O(2^{2^{|\phi|}|\phi|} \cdot |\phi|^3 \cdot B_{|\phi|} \cdot E_{\max}^{|\phi|})$ in the size of the formula ϕ ;
- $O(2^{|Q_{\mathcal{H}}|^2} \cdot |Q_{\mathcal{H}}|^4)$ in the size of the model \mathcal{H} ;
- $O(E_{\max}^{|\phi|})$, i.e., polynomial in the largest number of entities in \mathcal{H} .

Discussion. Clearly, here the presence of entities in the states and the valuations that must be taken into account produces a rather expensive overhead with respect to the tableau algorithm for LTL presented in [17] that is only exponential in $|\phi|$. However, as remarked in [17] most interesting properties about safety and liveness can be formulated by small size formulae.

Concerning the parameter E_{\max} , in order to have an idea how it may grow, we can consider the programming language \mathcal{L} of Section 4. There E_{\max} is bounded by the number of variables in the program.

Although, the bound given here is rather rough, more problematic is the complexity with respect to size of the model since in principle nothing can be assumed for the cardinality of $Q_{\mathcal{H}}$. The LTL algorithm is linear in the size of the model (instead of exponential) because it uses *maximal* strong connected components (MSCC) instead of SCS. It is very likely that the same optimisation would provide similar benefits in the algorithm defined in this section. Nevertheless, in this paper the emphasis is in the decidability result while optimisations of the algorithm is postponed to future work.

6 Related work

History-dependent automata History-dependent (HD) automata [20] are the main inspiration for HABAs. An HD-automaton is an automaton where states, transitions and labels are equipped with a set of local names that can be created dynamically. HD-automata represent an adequate model for history-dependent formalisms such as the π -calculus. Reallocation of entities in HABA resembles the reallocation of names in HD-automata. HD-automata and HABAs differ in the way in which entities are referred to. In a HABA, entities can only be addressed by means of logical variables that can be compared by ATL-formulae. More important, though, is the novelty introduced in HABAs by the black hole abstraction. This key feature allows us to deal with a possibly unbounded number of entities.

Spatial logic Related to ATL, concerning properties of freshness, is the recent Spatial Logic (SL) [5, 4]. SL is defined for the Ambient Calculus and has modalities that refer to space as well as time. Freshness can be identified in SL using a special quantifier, and has a somewhat different interpretation than in ATL. More precisely, in SL “fresh” means distinct from any name used in the formula and in the model satisfying it. If there is a fresh name, there are infinitely many of them (Gabbay-Pitts property [13]). In contrast, in ATL, if an entity is fresh it means that the entity is used in the current state and did not appear previously. This conceptual difference has several consequences. For instance, there exist non-contradictory ATL-formulae where more than one distinct fresh entity is identified in the same state. Another difference between SL and ATL concerns quantification. In SL, quantification is over a fixed (countable) set of names, whereas in ATL, quantification ranges over entities that are alive in the current state. This set is not fixed from state to state. Therefore, e.g., the ATL formula $\forall x.X\phi$ is not equivalent to $X\forall x.\phi$.

Tableau-based methods There are basically two approaches to model-checking temporal logics: the automata-theoretic approach (for LTL [25] and CTL [10, 16]) and the tableau method. Tableaux are typically used for the solution of more general problems, like satisfiability. For model checking, the tableau approach was first developed for CTL [6, 2]. Our algorithm is strongly based on the tableau method for LTL presented in [17].

Model-checking and logics for object-oriented systems Model-checking tools for object-oriented systems are becoming more and more popular, but the property specification formalisms are not tailored towards properties over objects (such as allocation and de-allocation). Bandera [8] is a model checker for Java that uses abstract interpretation and program slicing to yield compact state spaces. Another model checker for Java is Java

PathFinder [14]. JPF employs garbage collection in order to obtain a finite state space. However, in the approaches we are aware of, dynamic creation of objects is only supported to a limited extent (the number of created objects must be bounded). Both Bandera and JPF use LTL (or CTL) as property specification language.

The recent paper [27] is related to our work since it deals with unbounded number of Java objects and threads. The approach is mostly based on abstract interpretation. The main idea is to conservatively represent (via 3-valued logical structures) many configurations using a single abstract configuration. This clearly relates with the black hole abstraction. However, the paper deals only with *safety* properties and in particular, the emphasis is on interference between Java thread, deadlock, shared abstract data types, and illegal thread interactions. Furthermore, this technique may *falsely* report that a safety property may be violated, although it can never miss a violation. This is in contrast with our approach that always provides correct answers.

Apart from these (tool-oriented) approaches, several temporal logics for object-oriented systems have been defined [15, 23, 12, 9], that, however, do not support primitives for the birth and death of objects; most of these logic rely on higher order logics to deal with class hierarchies and are aimed at theorem proving.

7 Conclusions and future work

In this paper we presented an extension of LTL that deals with the notions of *dynamic allocation* and *deallocation*. The new logic, called Allocational Temporal Logic, expresses interesting properties for a wide range of applications. An example is object-oriented programming, where creation/destruction of objects are key issues. In a more abstract level, allocation and deallocation of *resources* (channels, keys, processes, etc.) are fundamental concepts in many fields of computer science.

We defined two different extensions of (generalised) Büchi automata, called ABA and HABA, whose runs generate models of ATL-formulae. ABAs are mostly *infinite state* while HABAs are a symbolic representation of ABAs and therefore are mostly *finite state*. The two main ingredients to obtain finite state HABAs are *reallocations*, firstly introduced in [20] for History Dependent Automata, and *black-hole abstraction*. The latter allows us to deal with certain cases of unboundness.

As an example of the application of ABAs and HABAs theory, we introduced a simple programming language dealing with the creation and destruction of abstract entities. Two operational semantics of the language were given: a concrete semantics (in terms of ABA) and a symbolic one (in terms of HABA). The symbolic semantics always produces finite models. The two semantics were proved to be equivalent. Although the language is quite simple, we hope that this result can be extended to more realistic languages. Probably, more sophisticated versions of black-hole abstraction must be defined, e.g., with *multiple* black holes.

As a main result, we proved that the model checking problem for ATL is decidable. In particular, we defined in Section 5 an algorithm for model checking ATL-formulae that generalises the tableau method of Lichtenstein and Pnueli for LTL [17]. To the best of our knowledge, this represents the first effective approach for model-checking models with an unbounded number of entities.

Future work In the future we plan to investigate the use of HABA for the definition of the semantics of more realistic OOP languages. The first step would be the definition of automata where entities can reference each other.

Another (long term) open research question is the satisfiability of ATL. During the design of model checking strategies for ATL, it turned out that our current definition of HABAs is not suitable for an automata based approach to model checking. The definition of HABAs, although appropriate for automatic verification, is not expressive enough to provide canonical models for ATL-formulae.

Similarly, it would be interesting to extend ATL in a way that it would be possible to express properties of program variables in the language \mathcal{L} . The development of a proof theory for ATL should also be investigated.

Finally, lifting ATL to a second order logic would be an appealing enrichment in the expressiveness of the formalism. In fact, this would allow to express not only that entities are new with respect to the previous state, but more specifically, with respect to some particular state in the past.

References

- [1] J. W. de Bakker, E. de Vink. Control Flow Semantics. MIT Press, Cambridge, MA, 1996.
- [2] M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Inf.*, 20(3):207–226, 1983.
- [3] J. Büchi. Weak second order logic and finite automata. *Z. Math. Logik, Grundlag. Math.*,5:66–62,1960.
- [4] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). In *TACS'01*, LNCS 2255, pp. 1–37, Springer, 2001.
- [5] L. Cardelli and A. D. Gordon. Logical properties of name restriction. In *TLCA'01*, LNCS 2044, pp. 46–60, Springer, 2001.
- [6] E. Clarke and E. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logics of Programs*, LNCS 131, pp. 52–71, Springer, 1981.
- [7] E. Clarke, O. Grumberg, D.A. Peled. Model Checking. MIT Press, Cambridge, MA, 1999.
- [8] J. Corbett, M. Dwyer, J. Hatcliff, C. Pasareanu, Robby, S. Laubach, and H. Zheng. Bandera: Extracting finite-state models from Java source code. In *ICSE'00*, pp. 439–448, IEEE CS Press, 2000.
- [9] D. Distefano, J.-P. Katoen, and A. Rensink. On a temporal logic for object-based systems. In *FMOODS'00*, pp. 305–326, Kluwer, 2000.
- [10] E. A. Emerson. Automata, tableaux and temporal logics. In *Logic of Programs*, LNCS 193, pp. 79–88, Springer, 1985.

- [11] G. Ferrari, G. Ferro, S. Gnesi, U. Montanari, M. Pistore, and G. Ristori. An automata based verification environment for mobile processes. In *TACAS97*, LNCS 1217, pp. 275–289, 1997.
- [12] J. Fiadeiro and T. Maibaum. Verifying for reuse: foundations of object-oriented system verification. In *Theory and Formal Methods*, pages 235–257, 1995.
- [13] M. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In *LICS'99*, pages 214–224, IEEE CS Press, 1999.
- [14] K. Havelund and T. Pressburger. Model checking Java programs using Java PathFinder. *Int. J. on Software Tools for Technology Transfer*, 2(4):366–381, 2000.
- [15] R. Jungclaus, G. Saake, T. Hartmann, and C. Sernadas. TROLL: A language for object-oriented specification of information systems. *ACM Tr. on Inf. Sys.*, 14(2):175–211, 1996.
- [16] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *J. of the ACM*, 47(2):312–360, 2000.
- [17] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *POPL'85*, pp. 97–107, ACM Press, 1985.
- [18] Z. Manna, A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.
- [19] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (part I and II). *Inf. & Comp.*, 100(1): 1-77, 1992.
- [20] U. Montanari and M. Pistore. An introduction to history-dependent automata. *Electr. Notes in Th. Comp. Sci.*, 10, 1998.
- [21] A. M. Odlyzko. Asymptotic enumeration methods. In R. Graham *et al.* (eds.), *Handbook of Combinatorics*, vol. 2, chapter 22, pp. 1063–1229, 1995.
- [22] A. Pnueli. The temporal logic of programs. In *FOCS'77*, pp. 46–57, IEEE CS Press, 1977.
- [23] A. Sernadas, C. Sernadas, and J.F. Costa. Object specification logic. *J. of Logic and Computation*, 5(5):603–630, 1995.
- [24] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pp. 133-191. Elsevier Science Publishers B. V., 1990.
- [25] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *LICS'86*, pp. 332–344, IEEE CS Press, 1986.
- [26] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logic of programs. *J. of Computer and System Sciences*, 32:183–221, 1986.
- [27] E. Yahav. Verifying safety properties of concurrent Java programs using 3-valued logic. In *POPL 2001*, pp. 27-40 ACM Press, 2001.

A Proofs of Section 2

Proposition 2.9 For ATL-formula ϕ , folded allocational sequences σ_f and allocational sequence σ_u :

1. For every $(\sigma_u, N_u, \theta_u)$ there exists a $(\sigma_f, N_f, \theta_f)$ such that $(\sigma_u, N_u, \theta_u) \sqsubseteq^{fold} (\sigma_f, N_f, \theta_f)$
2. For every $(\sigma_f, N_f, \theta_f)$ there exists a $(\sigma_u, N_u, \theta_u)$ such that $(\sigma_u, N_u, \theta_u) \sqsubseteq^{fold} (\sigma_f, N_f, \theta_f)$.

Proof.

1. Trivial by taking $(\sigma_f, N_f, \theta_f) = (id(\sigma_u), N_u, \theta_u)$.
2. Let $\sigma_f = E_0 \lambda_0 E_1 \lambda_1 \dots$. We prove the proposition by defining $(\sigma_u, N_u, \theta_u)$ such that $id(\sigma_u) \cong \sigma_f$. Let $\sigma_u = E'_0 E'_1 \dots$. By definition of \cong , $id(\sigma_u) \cong \sigma_f$ if (for all $i \geq 0$) $E_i \cong E'_i$ and

$$\lambda_i \circ h_i = h_{i+1} \circ id_i \quad (6)$$

where h_i is an isomorphism between E_i and E'_i .

We show that it is possible to define such E'_i by induction on i .

[Base] It suffices to choose E'_0 such that $|E_0| = |E'_0|$. As these sets are isomorphic, let h_0 be such isomorphism between E_0 and E'_0 .

[Step] Assume we have constructed the sequences up to index i (with $i > 0$). The proof obligation is to construct $E'_{i+1} \cong E_{i+1}$ satisfying (6). Let $E'_{i+1} = E_{old} \cup E_{new}$ with $E_{old} \cap E_{new} = \emptyset$. Choose $E_{old} = h_i^{-1}(\text{dom}(\lambda_i))$ i.e., $E_{old} \subseteq E'_i$ is the set of entities that do not die during the transition from E_i to E_{i+1} , and that in E'_{i+1} are old. E_{new} corresponds to the new entities of E_{i+1} . Choose E_{new} such that $|E_{new}| = |E_{i+1} \setminus \text{cod}(\lambda_i)|$ and $E_{new} \cap E'_i = \emptyset$. The first constraint on E_{new} avoids to choose entities that in E'_i correspond to entities in E_i that died during the transition from E_i to E_{i+1} . The second constraint establishes $E_{old} \cap E_{new} = \emptyset$. As $|E_{new}| = |E_{i+1} \setminus \text{cod}(\lambda_i)|$, E_{new} is isomorphic to the set of new entities in E_{i+1} ; let $h : E_{new} \rightarrow (E_{i+1} \setminus \text{cod}(\lambda_i))$ be such isomorphism. Then we define $h_{i+1} : E'_{i+1} \rightarrow E_{i+1}$ by:

$$h_{i+1}(e) = \begin{cases} \lambda_i \circ h_i(e) & \text{if } e \in E_{old} \\ h(e) & \text{if } e \in E_{new}. \end{cases}$$

It is easy to see that this definition satisfies (6). □

B Proofs of Section 3

Lemma 3.9 For HABA \mathcal{H} and any expansion $Exp(\mathcal{H})$ of \mathcal{H} :

- (a) $\mathcal{L}(Exp(\mathcal{H})) \sqsupseteq^{fold} \mathcal{L}(\mathcal{H})$ and
- (b) $\mathcal{L}(Exp(\mathcal{H})) \sqsubseteq^{fold} \mathcal{L}(\mathcal{H})$.

Proof. Let \mathcal{H} be a HABA and $Exp(\mathcal{H})$ an ABA that expands \mathcal{H} with $\psi : Q_{Exp(\mathcal{H})} \rightarrow Q_{\mathcal{H}}$ surjective.

(a) Let $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{H})$. To prove: there exists $(\sigma', N', \theta') \in \mathcal{L}(Exp(\mathcal{H}))$ such that $(\sigma', N', \theta') \sqsubseteq^{fold} (\sigma, N, \theta)$. Let $\sigma = E_0 \lambda_0^\sigma E_1 \lambda_1^\sigma \dots$ be such that $\rho = q_0 \lambda_0 q_1 \lambda_1 \dots \in \text{runs}(\mathcal{H})$ generates (σ, N, θ) . Let $\rho' = q'_0 \lambda'_0 q'_1 \lambda'_1 \dots \in \text{runs}(Exp(\mathcal{H}))$. As $Exp(\mathcal{H})$ expands \mathcal{H} , it follows that for $i \geq 0$: $q_i = \psi(q'_i)$, $q'_i \rightarrow q'_{i+1}$ expands $q_i \rightarrow_{\lambda_i} q_{i+1}$ and $|E_{q'_i}| = |E_{q_i}|$. These facts relate ρ , ρ' and σ . Let $\sigma' = E_{q'_0} \lambda_{q'_0} E_{q'_1} \lambda_{q'_1} \dots$. It follows directly that ρ' accepts σ' . Note that, by construction, $\phi_{q'_i} : E_{q'_i} \rightarrow E_{q_i}$ and $\phi_i : E_i \rightarrow E_{q_i}^\infty$ are bijective on the entities that are not mapped onto ∞ , i.e., $|E_i \setminus E_i^\infty| = |E_{q_i}| = |E_{q'_i} \setminus E_{q'_i}^\infty|$ for all $i \geq 0$. In order to show that $id(\sigma') \cong \sigma$, we prove that there exists a family of bijections $(h_i)_{i \geq 0}$ with $h_i : E_{q'_i} \rightarrow E_i$ such that

$$\lambda_i^\sigma \circ h_i = h_{i+1} \circ id_{E_{q'_i} \cap E_{q'_{i+1}}} . \quad (7)$$

Then it follows $(id(\sigma'), h_0^{-1}(N), h_0^{-1} \circ \theta) \cong (\sigma, N, \theta)$, and by definition, $(\sigma', h_0^{-1}(N), h_0^{-1} \circ \theta) \sqsubseteq^{fold} (\sigma, N, \theta)$, which proves (a). Consider the following definitions of h_i . For $i = 0$, let:

$$h_0(e) = \begin{cases} \tilde{h}(e) & \text{if } \phi_{q'_0}(e) = \infty \\ \phi_0^{-1} \circ \phi_{q'_0}(e) & \text{if } \phi_{q'_0}(e) \neq \infty \end{cases}$$

where \tilde{h} is an arbitrary isomorphism between $E_{q'_0}^\infty$ and E_0^∞ . Note that such isomorphism exists, since $|E_{q'_i}^\infty| = |E_i^\infty|$ for all i . For $i > 0$, let:

$$h_{i+1}(e) = \begin{cases} \lambda_i^\sigma \circ h_i \circ id_{E_{q'_i} \cap E_{q'_{i+1}}}^{-1}(e) & \text{if } \phi_{q'_{i+1}}(e) = \infty \\ \phi_{q'_{i+1}}^{-1} \circ \phi_{q'_{i+1}}(e) & \text{if } \phi_{q'_{i+1}}(e) \neq \infty \end{cases}$$

The fact that h_{i+1} is well defined can be seen as follows. If $\phi_{q'_{i+1}}(e) = \infty$ then $e \in E_{q'_i} \cap E_{q'_{i+1}}$, otherwise the number of new entities in $E_{q'_{i+1}}$ and $E_{q'_{i+1}}$ would differ, which cannot be the case due to condition (ii) of Def. 3.8. Thus, $id_{E_{q'_i} \cap E_{q'_{i+1}}}^{-1}(e)$ is defined. If $\phi_{q'_{i+1}}(e) \neq \infty$ then $h_{i+1}(e)$ is defined, since $\phi_{q'_{i+1}}$ is bijective on entities not mapped onto ∞ .

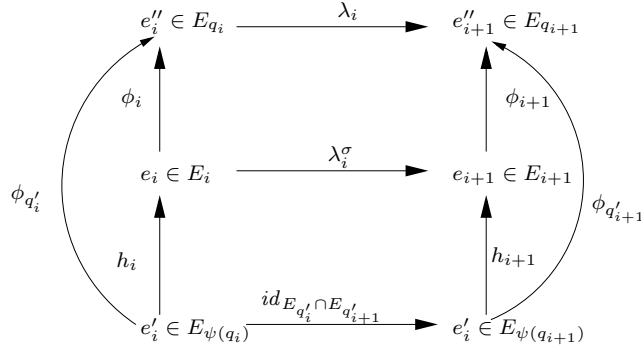


Figure 17: q_i is generated by $q'_i = \psi(q_i)$.

We now show that h_i satisfies (7) in the following steps (cf. Fig. 17): For all $i \geq 0$:

$$\phi_{q'_i} = \phi_i \circ h_i \tag{8}$$

The proof is by induction on i .

(Base) For $i = 0$ it follows directly from the definition of h_0 .

(Step) Assume (8) holds for $i > 0$. We prove case $i+1$. According to the definition of h_{i+1} , we distinguish:

- $\phi_{q'_{i+1}}(e) \neq \infty$. Then $\phi_{i+1} \circ h_{i+1}(e) = \phi_{i+1} \circ \phi_{q'_{i+1}}^{-1} \circ \phi_{q'_{i+1}}(e) = \phi_{q'_{i+1}}(e)$.
- $\phi_{q'_{i+1}}(e) = \infty$. Then we have

$$\begin{aligned} & \phi_{i+1} \circ h_{i+1}(e) \\ &= \phi_{i+1} \circ \lambda_i^\sigma \circ h_i \circ id_{E_{q'_i} \cap E_{q'_{i+1}}}^{-1}(e) \\ &= [\phi_{q'_{i+1}}(e) = \infty \text{ implies } e \in E_{q'_i} \cap E_{q'_{i+1}}] \\ & \quad \phi_{i+1} \circ \lambda_i^\sigma \circ h_i(e) \\ &= [\text{by condition 4 of Def. 3.6}] \\ & \quad \lambda_i \circ \phi_i \circ h_i(e) \\ &= [\text{by induction hypothesis}] \\ & \quad \lambda_i \circ \phi_{q'_i} \\ &= [\text{by condition (i) of Def. 3.8}] \\ & \quad \phi_{q'_{i+1}} \circ id_{E_{q'_i} \cap E_{q'_{i+1}}}(e) \\ &= \phi_{q'_{i+1}}(e) \end{aligned}$$

This completes the proof of (8). Using this fact, we prove for all $i \geq 0$:

$$\text{dom}(\lambda_i^\sigma \circ h_i) = \text{dom}(h_{i+1} \circ \text{id}_{E_{q'_i} \cap E_{q'_{i+1}}}) \quad (9)$$

Note that $\text{dom}(h_{i+1} \circ \text{id}_{E_{q'_i} \cap E_{q'_{i+1}}}) = E_{q'_i} \cap E_{q'_{i+1}}$ and $\text{dom}(\lambda_i^\sigma \circ h_i) = \{e \in E_{q'_i} \mid h_i(e) \in \text{dom}(\lambda_i^\sigma)\}$.

- ‘ \supseteq ’: let $e \in E_{q'_i} \cap E_{q'_{i+1}}$. By (8) it follows $\phi_{q'_i}(e) = \phi_i \circ h_i(e)$. By condition (i) of Def. 3.8, we have that $\lambda_i \circ \phi_i \circ h_i(e)$ is defined, since $e \in E_{q'_i} \cap E_{q'_{i+1}}$. This implies by condition 4 of Def. 3.6 that $\phi_{i+1} \circ \lambda_i^\sigma \circ h_i(e)$ is defined and therefore $e \in \text{dom}(\lambda_i^\sigma \circ h_i)$.
- ‘ \subseteq ’: let $e \in \text{dom}(\lambda_i^\sigma \circ h_i)$. Since $\lambda_i^\sigma \circ h_i(e)$ is defined, $\phi_{i+1} \circ \lambda_i^\sigma \circ h_i(e)$ is defined as well. By condition 4 of Def. 3.6, $\lambda_i \circ \phi_i \circ h_i(e)$ is defined. From (8) it follows that $\lambda_i \circ \phi_{q'_i}(e)$ is defined. But then by condition (i) of Def. 3.8, it must be $e \in E_{q'_i} \cap E_{q'_{i+1}}$.

This completes the proof of (9). Finally, we prove for all $i \geq 0$:

$$\forall e \in \text{dom}(\lambda_i^\sigma \circ h_i) : \left(\lambda_i^\sigma \circ h_i(e) = h_{i+1} \circ \text{id}_{E_{q'_i} \cap E_{q'_{i+1}}}(e) \right). \quad (10)$$

The equality can be also understood by the fact that the diagram in Fig. 17 commutes. Let $e \in E_{q'_i} \cap E_{q'_{i+1}}$. We distinguish two cases:

- $\phi_{q'_{i+1}}(e) \neq \infty$. Then

$$\begin{aligned} & \lambda_i^\sigma \circ h_i(e) \\ &= [\text{by definition of } h_i] \\ & \lambda_i^\sigma \circ \phi_i^{-1} \circ \phi_{q'_i}(e) \\ &= [\text{since } \phi_{i+1}^{-1}(e) \text{ is defined and by Def. 3.6 } \phi_{i+1}^{-1} \circ \lambda_i \circ \phi_i = \lambda_i^\sigma] \\ & \phi_{i+1}^{-1} \circ \lambda_i \circ \phi_i \circ \phi_i^{-1} \circ \phi_{q'_i}(e) \\ &= \phi_{i+1}^{-1} \circ \lambda_i \circ \phi_{q'_i}(e) \\ &= [q'_i \rightarrow q'_{i+1} \text{ expands } q_i \rightarrow_{\lambda_i} q_{i+1}, \text{ by condition (i) of Def. 3.8}] \\ & \phi_{i+1}^{-1} \circ \phi_{q'_{i+1}} \circ \text{id}_{E_{q'_i} \cap E_{q'_{i+1}}}(e) \\ &= [\phi_{q'_{i+1}}(e) \neq \infty, \text{ definition of } h_{i+1}] \\ & h_{i+1} \circ \text{id}_{E_{q'_i} \cap E_{q'_{i+1}}}(e). \end{aligned}$$

- $\phi_{q'_{i+1}}(e) = \infty$. Then we have

$$\begin{aligned} & \lambda_i^\sigma \circ h_i(e) \\ &= \lambda_i^\sigma \circ h_i \circ \text{id}_{E_{q'_i} \cap E_{q'_{i+1}}}^{-1} \circ \text{id}_{E_{q'_i} \cap E_{q'_{i+1}}}(e) \\ &= [\text{by definition of } h_{i+1}] \\ & h_{i+1} \circ \text{id}_{E_{q'_i} \cap E_{q'_{i+1}}}(e). \end{aligned}$$

From (9) and (10) it follows (7). This completes the proof of (a).

(b) Let $(\sigma, N, \theta) \in \mathcal{L}(\text{Exp}(\mathcal{H}))$ with $\sigma = E_0 E_1 E_2 \dots$ and $\rho = q_0 q_1 q_2 \dots$ be the run that generates σ . To prove: there exists $\sigma' \in \mathcal{L}(\mathcal{H})$ such that $\sigma' \cong \text{id}(\sigma)$. Since $\rho \in \text{runs}(\text{Exp}(\mathcal{H}))$, there exists $\rho' = \psi(q_0) \lambda_0 \psi(q_1) \lambda_1 \dots \in \text{runs}(\mathcal{H})$ such that $q_i \rightarrow q_{i+1}$ expands $\psi(q_i) \rightarrow_{\lambda_i} \psi(q_{i+1})$ for all $i \geq 0$. Since ρ is a run of $\text{Exp}(\mathcal{H})$, every state in ρ satisfies conditions 1–6 of Def. 3.8 for a family of functions $\phi_{q_i} : E_{q_i} \rightarrow E_{\psi(q_i)}^\infty$. Choose $\sigma' = \text{id}(\sigma)$. We show that ρ' generates $(\text{id}(\sigma), N, \theta)$. This amounts to prove the existence of a generator $\phi'_i : E_i \rightarrow E_{\psi(q_i)}^\infty$ satisfying the conditions of Def. 3.6. Let $\phi'_i = \phi_{q_i}$. Conditions 1–3 follow directly from the definition of ϕ_{q_i} (since by definition it satisfies conditions 1–3 of Def. 3.8). Furthermore, by definition of expansion: $\lambda_i \circ \phi_{q_i} = \phi_{q_{i+1}} \circ \text{id}_{E_{q_i} \cap E_{q_{i+1}}}$. Thus, condition 4 is fulfilled. Consider now condition 5. Assume $e \in E_{i+1}$, $\phi_{i+1}(e) = \infty$ and $e \notin E_i$. Then $e \in E_{q_{i+1}} \setminus E_{q_i}$ and therefore $\phi_{i+1}(e) \in E_{\psi(q_{i+1})} \setminus \text{cod}(\lambda_i)$. This is, however, impossible since by hypothesis $\phi_{i+1}(e) = \infty$. Finally, condition 6 holds by definition of ϕ_i . Hence, we conclude that $(\text{id}(\sigma), N, \theta) \in \mathcal{L}(\mathcal{H})$. \square

Theorem 3.10 For any HABA \mathcal{H} and expansion $Exp(\mathcal{H})$ of $\mathcal{H} : \mathcal{L}(\mathcal{H}) \cong \mathcal{L}(id(Exp(\mathcal{H})))$.

Proof. ‘ \sqsubseteq^{fold} ’: Let $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{H})$. To prove: there exists $(\sigma', N', \theta') \in \mathcal{L}(id(Exp(\mathcal{H})))$ such that $(\sigma, N, \theta) \cong (\sigma', N', \theta')$. By Lemma 3.9.a it follows that there exists $(\sigma'', N'', \theta'') \in \mathcal{L}(Exp(\mathcal{H}))$ such that $(\sigma, N, \theta) \sqsubseteq^{fold} (\sigma'', N'', \theta'')$. By definition of \sqsubseteq^{fold} , thus $(\sigma, N, \theta) \cong (id(\sigma''), N'', \theta'') \in \mathcal{L}(id(Exp(\mathcal{H})))$.

‘ \sqsupseteq^{fold} ’: Let $(\sigma, N, \theta) \in \mathcal{L}(id(Exp(\mathcal{H})))$. To prove: there exists $(\sigma', N', \theta') \in \mathcal{L}(\mathcal{H})$ such that $(\sigma, N, \theta) \cong (\sigma', N', \theta')$. As $(\sigma, N, \theta) \in \mathcal{L}(id(Exp(\mathcal{H})))$ we have $(\sigma, N, \theta) = (id(\sigma''), N, \theta)$ for some σ'' with $(\sigma'', N, \theta) \in \mathcal{L}(Exp(\mathcal{H}))$. By Lemma 3.9.b it follows that $(\sigma'', N, \theta) \sqsubseteq^{fold} (\sigma', N', \theta')$ for some $(\sigma', N', \theta') \in \mathcal{L}(\mathcal{H})$. By definition of \sqsubseteq^{fold} , thus $(\sigma, N, \theta) \cong (\sigma', N', \theta')$. \square

C Proofs of Section 4

Theorem 4.5 For any $p \in \mathcal{L}$: \mathcal{A}_p is an expansion of \mathcal{H}_p .

Proof. Let $\psi : Q_{\mathcal{A}_p} \rightarrow Q_{\mathcal{H}_p}$ be defined in the following way:

$$\psi(s, E, \gamma) = (s, E') \text{ where } E' = \{\gamma^{-1}(e) \mid e \in E\}.$$

If entity $e \in E$ is not referenced by any variable, then $\gamma^{-1}(e) = \emptyset$. Thus, $\emptyset \in E'$ and state $\psi(s, E, \gamma)$ is unbounded. For state $q \in Q_{\mathcal{A}_p}$, let $\phi_q : E_q \rightarrow E_{\psi(q)}$ be defined as follows:

$$\phi_{(s, E, \gamma)}(e) = \gamma^{-1}(e).$$

To show that \mathcal{A}_p expands \mathcal{H}_p we prove that ϕ_q fulfills conditions 1–6 of Def. 3.8:

1. If $\phi_{(s, E, \gamma)}(e) = \phi_{(s, E, \gamma)}(e') \neq \emptyset$, then $\gamma^{-1}(e) = \gamma^{-1}(e')$. Since γ is well defined, it follows $e = e'$.
2. Straightforward, since if $X_i \in E_{\psi(q)}$, then there exists e such that $X_i = \gamma^{-1}(e)$.
3. If $\emptyset \notin E_{\psi(q)}$ then, by definition of ψ , we have for all $e \in E : e \in \text{cod}(\gamma)$. Therefore, by definition of ϕ_q , we have for all $e \in E : \phi_q(e) \neq \emptyset$.

5.+6. Follows directly, as the set of initial and final states for \mathcal{H}_p and \mathcal{A}_p are identical.

It remains to prove the 4th condition:

- a) any $\psi(s_1, E_1, \gamma_1) \rightarrow_\lambda q'_2$ is expanded by some transition $(s_1, E_1, \gamma_1) \rightarrow (s_2, E_2, \gamma_2)$, and
- b) any $(s_1, E_1, \gamma_1) \rightarrow (s_2, E_2, \gamma_2)$ is expanded by some $\psi(s_1, E_1, \gamma_1) \rightarrow_\lambda \psi(s_2, E_2, \gamma_2)$ for some λ .

These statements are proven by induction on the structure of statement s_1 . The base cases:

- **case new(v).** Let $q_1 = (\text{new}(v), E_1, \gamma_1) \in Q_{\mathcal{A}_p}$. We have that $\psi(q_1) = (\text{new}(v), E_{\psi(q_1)})$. According to the symbolic semantics (cf. Table 2), the only possible transition is

$$\text{new}(v), E_{\psi(q_1)} \rightarrow_\lambda \text{skip}, E' \tag{11}$$

where $E' = \{X_i \setminus \{v\} \mid X_i \in E_{\psi(q_1)}\} \cup \{\{v\}\}$. The only transition of q_1 (cf. Table 1) is

$$\text{new}(v), E_1, \gamma_1 \rightarrow \text{skip}, E_2, \gamma_1 \{e/v\} \tag{12}$$

where $E_2 = E_1 \cup \{e\}$ and $e \notin E_1$. For convenience, let $q_2 = \text{skip}, E_2, \gamma_1 \{e/v\}$. We show that (12) expands (11). First, we observe that $\psi(q_2) = (\text{skip}, E_{\psi(q_2)}) = (\text{skip}, E')$, since:

$$\begin{aligned} & E_{\psi(q_2)} \\ &= \{\gamma_1 \{e/v\}^{-1}(e') \mid e' \in E_1 \cup \{e\}\} \\ &= \{\gamma_1 \{e/v\}^{-1}(e') \mid e' \in E_1\} \cup \{\gamma_1 \{e/v\}^{-1}(e)\} \\ &= [\text{since } e \notin E_1] \\ & \quad \{\gamma_1^{-1}(e') \setminus \{v\} \mid e' \in E_1\} \cup \{\{v\}\} \\ &= \{X_i \setminus \{v\} \mid X_i \in E_{\psi(q_1)}\} \cup \{\{v\}\} \\ &= E'. \end{aligned}$$

Thus, $\psi(\text{skip}, E_1 \cup \{e\}, \gamma_1\{e/v\}) = (\text{skip}, E_{\psi(q_2)}) = (\text{skip}, E')$. Next, we need to check conditions (i) and (ii) of Def. 3.8. If $e' \in E_1$, we have:

$$\begin{aligned}
& \phi_{q_2}(id_{E_{q_1} \cap E_{q_2}}(e')) \\
&= [\text{since } e' \in E_{q_1} \Rightarrow e' \in E_{q_2}] \\
& \phi_{q_2}(e') \\
&= \gamma_1^{-1}\{e/v\}(e') \\
&= [\text{since } e' \in E_{q_1} \Rightarrow e' \neq e] \\
& \gamma_1^{-1}(e') \setminus \{v\} \\
&= [\text{by new's rule in Table 2}] \\
& \lambda(\gamma_1^{-1}(e')) \\
&= \lambda(\phi_{q_1}(e')).
\end{aligned}$$

This proves (i). (ii) follows directly from the fact that for **new** in both Table 2 and Table 1 only one new entity is created. Thus, transition (12) expands (11).

It remains to check condition 4.b. As we have seen above, the only possible transition $q_1 \rightarrow q_2$ in the concrete model is (12). Furthermore, we have $\psi(\text{new}(v), E_1, \gamma_1) = (\text{new}(v), E_{\psi(q_1)})$ and $\psi(\text{skip}, E_1 \cup \{e\}, \gamma_1\{e/v\}) = (\text{skip}, E_{\psi(q_2)}) = (\text{skip}, E')$ where $E' = \{X_i \setminus \{v\} \mid X_i \in E_{\psi(q_1)}\} \cup \{\{v\}\}$. Again, for $s_1 = \text{new}(v)$ the symbolic model prescribes only one transition, namely (11). As we have proved for case a), (12) expands (11).

- **case** $v := w$. Let $q_1 = (v := w, E_1, \gamma_1) \in Q_{\mathcal{A}_p}$, and $\psi(q_1) = (v := w, E_{\psi(q_1)})$. According to the rules of the symbolic model, $\psi(q_1)$ has only one possible transition (cf. Table 2), i.e., we have

$$v := w, E_{\psi(q_1)} \rightarrow_{\lambda} \text{skip}, E' \quad (13)$$

where $E' = \{X_i \setminus \{v\} \mid w \notin X_i\} \cup \{X_i \cup \{v\} \mid w \in X_i\}$. For q_1 there is only one possibility (cf. Table 1):

$$v := w, E_1, \gamma_1 \rightarrow \text{skip}, E_1, \gamma_1\{\gamma_1(w)/v\}. \quad (14)$$

We show that (14) expands (13):

$$\begin{aligned}
& E_{\psi(q_2)} \\
&= \{\gamma_1\{\gamma_1(w)/v\}^{-1}(e') \mid e' \in E_1\} \\
&= \{\gamma_1^{-1}(e') \setminus \{v\} \mid e' \in E_1 \setminus \{\gamma_1(w)\}\} \cup \\
& \quad \{\gamma_1^{-1}(\gamma_1(w)) \cup \{v\}\} \\
&= \{X_i \setminus \{v\} \mid w \notin X_i\} \cup \{X_i \cup \{v\} \mid w \in X_i\} \\
&= E'.
\end{aligned}$$

Note that if there is at least another variable $v' \neq v$ referring to v 's entity, then $\psi(q_2)$ remains bound if $\psi(q_1)$ was bound. If $\psi(q_1)$ was unbound, then $\psi(q_2)$ is unbound. Now, we show that conditions (i) and (ii) are fulfilled. Note that $e' \in E_{q_1} \Rightarrow e' \in E_{q_2}$. If $e' \neq \gamma_1(w)$ then $\phi_{q_2}(id_{E_{q_1} \cap E_{q_2}}(e')) = \phi_{q_2}(e') = \gamma_1^{-1}\{\gamma_1(w)/v\}(e') = \gamma_1^{-1}(e') \setminus \{v\} = \lambda(\gamma_1^{-1}(e')) = \lambda(\phi_{q_1}(e'))$. Similarly, if $e' = \gamma_1(w)$ then $\phi_{q_2}(id_{E_{q_1} \cap E_{q_2}}(e')) = \phi_{q_2}(e') = \gamma_1^{-1}\{\gamma_1(w)/v\}(e') = \gamma_1^{-1}(e') \cup \{v\} = \lambda(\gamma_1^{-1}(e')) = \lambda(\phi_{q_1}(e'))$. Note that there are no new entities both in the concrete and in the symbolic model, therefore (ii) holds.

The proof of condition 4.b follows in a straightforward manner from the above.

- **case** $\text{del}(v)$. Let $q_1 = (\text{del}(v), E_1, \gamma_1) \in Q_{\mathcal{A}_p}$ and $\psi(q_1) = (\text{del}(v), E_{\psi(q_1)})$. According to the rule for **del** in Table 2 we have

$$\text{del}(v), E_{\psi(q_1)} \rightarrow_{\lambda} \text{skip}, E_{\psi(q_1)} \setminus \{X_i\}. \quad (15)$$

where $X_i = \gamma_1^{-1}(\gamma_1(v))$. In the concrete model we have

$$\text{del}(v), E_1, \gamma_1 \rightarrow \text{skip}, E_1 \setminus \{\gamma_1(v)\}, \gamma_1\{\perp/v\}. \quad (16)$$

We show that there is correspondence between the target states of these transitions.

$$\begin{aligned}
& E_{\psi(q_2)} \\
&= \{\gamma_1\{\perp/v\}^{-1}(e') \mid e' \in E_1 \setminus \{\gamma_1(v)\}\} \\
&= \{\gamma_1^{-1}(e') \mid e' \in E_1 \setminus \{\gamma_1(v)\}\} \\
&= \{\gamma_1^{-1}(e') \mid e' \in E_1\} \setminus \{\gamma_1^{-1}(\gamma_1(v))\} \\
&= E_{\psi(q_1)} \setminus \{X_i\}.
\end{aligned}$$

We can conclude that indeed $\psi(q_2) = (\text{skip}, E_{\psi(q_1)} \setminus \{X_i\})$. We must now show that conditions i) and ii) hold. If $e \in E_1$ and $e \neq \gamma_1(v)$ then $\phi_{q_2}(id_{E_{q_1} \cap E_{q_2}}(e)) = \phi_{q_2}(e) = \gamma_1^{-1}\{\perp/v\}(e) = \gamma_1^{-1}(e) = \lambda(\gamma_1^{-1}(e)) = \lambda(\phi_{q_1}(e))$. If $e = \gamma_1(v)$ then $e \notin E_1 \cap E_2$ therefore there is nothing to prove because $e \notin \text{dom}(\phi_{q_2} \circ id_{E_{q_1} \cap E_{q_2}})$ and $e \notin \text{dom}(\lambda \circ \phi_{q_1})$.

Furthermore, there is no new entity after the transition in either symbolic or the concrete model. Thus, we conclude that condition 4.a holds.

We prove condition 4.b for $s_1 = \text{del}(v)$. Consider the transition (16). We have seen that $\psi(q_1)$ has only one possible transition, namely (15) and we have proved that indeed the former transition is the expansion of the latter one using λ as defined by rule for `del` of the symbolic model. Hence we conclude that for the `del`(v) case, 4.b is fulfilled.

(Step) We only present the proof for sequential composition. The proofs for the other cases are conducted in a similar way and are omitted here. The case $s = \text{skip}; s_2$ is trivial. Assume $s = s_1; s_2$ and $q_1 = s_1; s_2, E_1, \gamma_1$ with $s_1 \neq \text{skip}$. We prove that any transition

$$\psi(s_1; s_2, E_1, \gamma_1) \rightarrow_\lambda s'_1; s_2, E_{\psi(q_2)} \quad (17)$$

is expanded by transition $s_1; s_2, E_1, \gamma_1 \rightarrow q_2$. Transition (17) is only possible if:

$$\psi(s_1, E_1, \gamma_1) \rightarrow s'_1, E_{\psi(q_2)}. \quad (18)$$

By the induction hypothesis, there exists a transition $s_1, E_1, \gamma_1 \rightarrow s'_1, E_2, \gamma_2$ that expands (18). Therefore, $s_1; s_2, E_1, \gamma_1 \rightarrow s'_1; s_2, E_2, \gamma_2$, that exists by the rules of Table 1, expands (17).

For condition 4.b, consider transition $s_1; s_2, E_1, \gamma_1 \rightarrow q_2$ in the concrete semantics. By the rule for sequential composition in Table 1 there must be a transition $s_1, E_1, \gamma_1 \rightarrow s'_1, E_2, \gamma_2$. By the induction hypothesis, this expands transition $\psi(s_1, E_1, \gamma_1) \rightarrow_\lambda s'_1, E_{\psi(q_2)}$. Thus, $s_1; s_2, E_1, \gamma_1 \rightarrow q_2$ expands $\psi(s_1; s_2, E_1, \gamma_1) \rightarrow_\lambda s'_1; s_2, E_{\psi(q_2)}$. \square

Proposition C.1. For all $q \in Q_{\mathcal{H}_p}$: $|E_q| \leq |PVar|$.

Proof. The set E_q is a partition of $A \subseteq PVar$. Thus, $|E_q| \leq |A| \leq |PVar|$. \square

Theorem 4.6 For any $p \in \mathcal{L}$: \mathcal{H}_p is finite state.

Proof. We show that

$$|Q_{\mathcal{H}_p}| \leq |s_{\max}|^m \cdot \left[1 + 2 \cdot \sum_{k=1}^{|PVar|} \binom{|PVar|}{k} B_k \right]$$

where B_k is the number of partitions of a set of k elements, $|s_{\max}|$ the size of the longest sequential statement in p and m the number of sequential components of p . In fact, since we are interested in the partition of subsets of $PVar$ that solely consist of defined variables, B_k corresponds to the number of partitions when k variables are defined. There are $\binom{|PVar|}{k}$ different ways to choose k distinct variables from the set $PVar$. Therefore, we have $\binom{|PVar|}{k} B_k$ partitions for k defined variables. Finally, we have to consider all possible k such that $0 < k \leq |PVar|$. The constant 2 considers the possibility to have a bounded or unbounded state. As all variables can be undefined (i.e., $E = \emptyset$), one has to be added. \square

D Proofs of Section 5

Lemma 5.3 $\mathcal{L}(\mathcal{H}) = \mathcal{L}(\mathcal{H}_\delta)$.

Proof. [$\mathcal{L}(\mathcal{H}) \subseteq \mathcal{L}(\mathcal{H}_\delta)$] Let $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{H})$, with $\sigma = E_0\lambda_0E_1\lambda_1\cdots$. Let $\rho = q_0\lambda_0q_1\lambda_1\cdots$ be the run generating (σ, N, θ) by some generator $(h_i)_{i \in \mathbb{N}}$. We define a run ρ' of \mathcal{H}_δ that generates (σ, N, θ) in the following way. Let $\rho' = q'_0\lambda_0q'_1\lambda_1\cdots$ such that $q'_0 = (q_0, h_0(N))$ and for all $i \geq 0$, $q'_{i+1} = (q_{i+1}, E_{q_{i+1}} \setminus \text{cod}(\lambda_i))$. According to the definition of duplication, we have $q'_i \in Q'$ for $i \geq 0$. Furthermore, since $q_i \rightarrow_{\lambda_i} q_{i+1}$ ($i \geq 0$) then there exists a corresponding transition $q'_i \rightarrow_{\lambda_i} q'_{i+1}$ in \mathcal{H}_δ . As q'_0 is an initial state of \mathcal{H}_δ and for every accept state q_i visited infinitely often, also the corresponding accept state q'_i is visited infinitely often, we conclude that $\rho' \in \text{runs}(\mathcal{H}_\delta)$. Finally, since $E_{q'_i} = E_{q_i}$ for all $i \geq 0$, and ρ' has the same ∞ -reallocations λ_i as ρ , then the generator $(h_i)_{i \in \mathbb{N}}$ generates (σ, N, θ) also from run ρ' . Hence we conclude that $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{H}_\delta)$.

[$\mathcal{L}(\mathcal{H}_\delta) \subseteq \mathcal{L}(\mathcal{H})$] Let $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{H}_\delta)$, with $\sigma = E_0\lambda_0E_1\lambda_1\cdots$, and let $\rho = q'_0\lambda_0q'_1\lambda_1\cdots$ be the run generating (σ, N, θ) . We define $\rho = q_0\lambda_0q_1\lambda_1\cdots$ such that for all $i > 0$, where $q'_i = (q_i, M_i)$. Since $\rho' \in \text{runs}(\mathcal{H}_\delta)$ implies $\rho \in \text{runs}(\mathcal{H})$ and ρ' generates (σ, N, θ) by a generator $(h_i)_{i \in \mathbb{N}}$ implies ρ generates (σ, N, θ) by the same $(h_i)_{i \in \mathbb{N}}$, we conclude that $(\sigma, N, \theta) \in \mathcal{L}(\mathcal{H})$. \square

In this section recall that $K(\phi) = \max\{|fv(\psi)| \mid \psi \in CL(\phi)\}$ and $\Omega(\alpha) = |\{a \in A \mid \alpha(a) = \infty\}|$. Furthermore, note that since $\Xi = \text{dom}(\Theta)$ for all valuations (ϕ, Ξ, Θ) , we can drop the Ξ -component without loss of information. Finally, in the following, from an arbitrary partial mapping $\theta: LVar \rightarrow Ent$ we derive a partition-based mapping partial $[\theta]: \mathbf{2}^{LVar} \rightarrow Ent$ as follows:

$$[\theta]: X \mapsto e \quad \text{if } X = \theta^{-1}(e).$$

The following lemma is an auxiliary result for the proof of Prop 5.15

Lemma D.1. For any path $\pi = (q_0, D_0, k_0)\lambda_0(q_1, D_1, k_1)\cdots$ and for any generator $(h_j)_{j \in \mathbb{N}}$ of allocational sequences generated by the underlying run ρ of π :

$$k_0 = \min(K(\phi), \Omega(h_0)) \text{ implies } \forall j \in \mathbb{N} : k_j = \min(K(\phi), \Omega(h_j)).$$

Proof. By inductive argument, assume that $k_j = \min(K(\phi), \Omega(h_j))$. By Definition 5.11, $k_{j+1} = \min(K(\phi), k_j + \Omega(\lambda_j))$. There are two cases. On the one hand, if $\lceil q_{j+1} \rceil$ then trivially we have $k_{j+1} = 0 = \Omega(h_{j+1}) = \min(K(\phi), \Omega(h_{j+1}))$.

On the other hand, assume $\lfloor q_{j+1} \rfloor$. Then it results $k_{j+1} \geq 0$ and we distinguish two further cases:

- if $k_j = K(\phi)$ then by inductive hypothesis $\Omega(h_j) \geq K(\phi)$ and $k_j + \Omega(\lambda_j) \geq K(\phi)$ that in turn implies, together with $\lfloor q_{j+1} \rfloor$ that $\Omega(h_{j+1}) \geq K(\phi)$ and $k_{j+1} = K(\phi)$. We conclude that $k_{j+1} = \min(K(\phi), \Omega(h_{j+1}))$.
- if $k_j = \Omega(h_j) < K(\phi)$ then by Definition 5.11 and by inductive hypothesis, $k_{j+1} = \min(K(\phi), k_j + \Omega(\lambda_j)) = \min(K(\phi), \Omega(h_j) + \Omega(\lambda_j)) = \min(K(\phi), \Omega(h_{j+1}))$, every imploded entity is preserved in the transition.

Hence, we conclude that $k_j = \min(K(\phi), \Omega(h_j))$ for all $j \geq 0$. \square

Lemma D.2. Let \mathcal{H} be a HABA and ϕ an ATL-formula. Let $\pi = (q_0, D_0, k_0)\lambda_0(q_1, D_1, k_1)\lambda_1\cdots$ be a path of $G_{\mathcal{H}}(\phi)$ with underlying run $\rho = q_0\lambda_0q_1\lambda_1\cdots$, and let σ be an allocational sequence generated by ρ with generator $(h_j)_{j \in \mathbb{N}}$, such that $k_j = \min(K(\phi), \Omega(h_j))$ for all $j \geq 0$. Then for all $i \geq 0$, $\psi \in CL(\phi)$ and $\theta: fv(\psi) \rightarrow Ent$:

$$\sigma^i, N_i^\sigma, \theta \models \psi \quad \iff \quad (\psi, h_i \circ [\theta]) \in D_i . \quad (19)$$

Proof. First of all we prove that there exists an allocational sequence σ generated by ρ with a generator $(h_j)_{j \in \mathbb{N}}$ such that $k_j = \min(K(\phi), \Omega(h_j))$ for all $j \geq 0$. By Lemma D.1, it is enough to show that there exists among the allocational sequences generated by ρ , one with $\Omega(h_0) = k_0$ (recall that $k_0 \leq K(\phi)$). We distinguish to cases:

- $[q_0]$ then $k_0 = 0$ and ρ generates only one sequence (up to isomorphism), i.e., precisely the one such that $\Omega(h_0) = 0$.
- $[q_0]$ then ρ generates every sequence with an arbitrary number of (initial) imploded entities. Clearly, among these sequences there exists a σ such that $\Omega(h_0) = k_0$.

Now, we can prove the statement (19) by induction on the structure of ψ .

Base of induction

- **case** $\psi = x \text{ new}$.

$[\Rightarrow]$ Suppose $\sigma^i, N_i^\sigma, \theta \models x \text{ new}$. This implies $x \in \text{dom}(\theta)$ and $\theta(x) = e$ for some $e \in N_i^\sigma$. By definition of the generator, $h_i(e) \in N_{q_i}$; therefore, $(x \text{ new}, \{x\} \mapsto h_i(e)) \in D_i$ by Defs. 5.5 and 5.8. Since $[\theta] = (\{x\} \mapsto e)$ we are done.

$[\Leftarrow]$ Suppose $(x \text{ new}, \Theta) \in D_i$ where $\Theta = h_i \circ [\theta]$. It follows that $x \in X \in \text{dom}(\Theta)$ such that $\Theta(X) \in N_{q_i}$; by the definition of generator and the construction of $[\theta]$ it follows that $\theta(x) = h_i^{-1}(\Theta(X)) \in N_i^\sigma$. Thus we have $\sigma^i, N_i^\sigma, \theta \models x \text{ new}$.

- **case** $\psi = (x = y)$.

$[\Rightarrow]$ Suppose $\sigma^i, N_i, \theta \models x = y$. This implies $x, y \in \text{dom}(\theta)$ and $\theta(x) = e = \theta(y)$ for some $e \in E_i^\sigma$; hence $[\theta] = (\{x, y\} \mapsto e)$. We have two cases:

1. $h_i(e) \neq \infty$. By definition of AV_{q_i} , we have $(x = y, \{x, y\} \mapsto h_i(e)) \in D_i$.
2. $h_i(e) = \infty$. Since $K(\phi) \geq K(\psi) = 2$, it follows that $k_i = \min(K(\phi), \Omega(h_i)) \geq 1$. Hence, by the definition of atoms (Definition 5.8), we have $(x = y, \{x, y\} \mapsto h_i(e)) \in D_i$.

$[\Leftarrow]$ Suppose $(x = y, \Theta) \in D_i$ with $\Theta = h_i \circ [\theta]$. It follows that $\text{dom}(\Theta) = \{\{x, y\}\}$ and $\Theta(\{x, y\}) = h_i(\theta(x)) = h_i(\theta(y))$; hence $x, y \in \text{dom}(\theta)$ and $\theta(x) = \theta(y)$ (by the construction of $[\theta]$). It follows that $\sigma^i, N_i^\sigma, \theta \models x = y$.

Inductive step

- **case** $\psi = \exists x.\psi'$.

$[\Rightarrow]$ Suppose $\sigma^i, N_i^\sigma, \theta \models \exists x.\psi'$. It follows that there exists an $e \in E_i^\sigma$ such that $\sigma^i, N_i^\sigma, \theta\{e/x\} \models \psi'$. By general assumption (see Page 26), $x \in \text{fv}(\psi')$ and hence $\text{dom}(\theta\{e/x\}) \subseteq \text{fv}(\psi')$; hence by the induction hypothesis, we have $(\psi', \Theta') \in D_i$ such that $\Theta' = h_i \circ [\theta\{e/x\}]$. Note that $x \in \bigcup \text{dom}(\Theta')$. It is not difficult to check that, for $\Theta = \Theta' \upharpoonright \text{fv}(\psi)$,

$$\Theta = \{(X \setminus \{x\}, \Theta'(X)) \mid X \in \text{dom}(\Theta'), X \neq \{x\}\} = h_i \circ [\theta]$$

(using in particular that $x \notin \text{dom}(\theta)$). It follows (by Definition 5.8 of atoms) that $(\exists x.\psi', \Theta) \in D_i$.

$[\Leftarrow]$ Suppose $(\exists x.\psi', \Theta) \in D_i$ such that $\Theta = h_i \circ [\theta]$. This implies by the definition of atom that $\exists(\psi', \Theta') \in D_i$ where

1. $\Theta = \Theta' \upharpoonright (\exists x.\psi')$ ($= \{(X \setminus \{x\}, \Theta'(X)) \mid X \in \text{dom}(\Theta'), X \neq \{x\}\}$);
2. $x \in \bigcup \text{dom}(\Theta')$;
3. $\Omega(\Theta') \leq k_i$.

We now want to construct $\theta' = \theta\{e/x\}$, in such a way that $\Theta' = h_i \circ [\theta']$. This comes down to choosing an appropriate e . There are three possibilities, based on $X \in \text{dom}(\Theta')$ such that $x \in X$:

- $\Theta'(X) \neq \infty$; then $e = \Theta'(X)$ is appropriate.
- $X \supset \{x\}$; then $e = \theta(y)$ for $y \in X \setminus \{x\}$ is appropriate.
- $\Theta'(X) = \infty$ and $X = \{x\}$. Then $\Omega(\Theta') = \Omega(\Theta) + 1$, hence $k_i \geq \Omega(\Theta) + 1$. It follows that $h_i(e) = \infty$ for some $e \notin \text{cod}(\theta)$; this e is appropriate.

By the induction hypothesis, it follows that $\sigma^i, N_i^\sigma, \theta' \models \psi'$. But then also $\sigma^i, N_i^\sigma, \theta \models \psi$.

- **case** $\psi = \neg\psi'$.

[\Rightarrow] Suppose $\sigma^i, N_i^\sigma, \theta \models \neg\psi'$. This implies $\sigma^i, N_i^\sigma, \theta \not\models \psi'$. By the induction hypothesis, it follows that $(\psi', h_i \circ [\theta]) \notin D_i$. But then (by the definition of atom) $(\neg\psi', h_i \circ [\theta]) \in D_i$.

[\Leftarrow] Inverse to the above.

- **case** $\psi = \psi_1 \vee \psi_2$.

[\Rightarrow] Suppose $\sigma^i, N_i^\sigma, \theta \models \psi_1 \vee \psi_2$. This implies either $\sigma^i, N_i^\sigma, \theta \models \psi_1$ or $\sigma^i, N_i^\sigma, \theta \models \psi_2$; w.l.o.g. assume the former. Let $\theta_1 = \theta \upharpoonright \text{fv}(\psi_1)$; it follows that also $\sigma^i, N_i^\sigma, \theta_1 \models \psi_1$. By the induction hypothesis, we have $(\psi_1, \Theta_1) \in D_i$ for $\Theta_1 = h_i \circ [\theta_1]$. Now let $\Theta = h_i \circ [\theta]$. Due to $\Omega(\Theta) \leq \text{fv}(\psi) \leq K(\phi)$, $\Omega(\Theta) \leq \Omega(h_i)$ and $k_i = \min(K(\phi), \Omega(h_i))$ we may conclude that $\Omega(\Theta) \leq k_i$. Since (as is easily checked) $\Theta_1 = \Theta \upharpoonright \psi_1$, we may conclude (by the definition of atom) that $(\psi_1 \vee \psi_2, \Theta) \in D_i$.

[\Leftarrow] Suppose $(\psi_1 \vee \psi_2, \Theta) \in D_i$ such that $\Theta = h_i \circ [\theta]$. By the definition of atom, this implies either $(\psi_1, \Theta \upharpoonright \psi_1) \in D_i$ or $(\psi_2, \Theta \upharpoonright \psi_2) \in D_i$; w.l.o.g. assume the former. Again, it is not difficult to see that $\Theta \upharpoonright \psi_1 = h_i \circ [\theta_1]$ where $\theta_1 = \theta \upharpoonright \text{fv}(\psi_1)$; hence by the induction hypothesis, $\sigma^i, N_i^\sigma, \theta_1 \models \psi_1$. But then also $\sigma^i, N_i^\sigma, \theta \models \psi_1$ and hence $\sigma^i, N_i^\sigma, \theta \models \psi_1 \vee \psi_2$.

- **case** $\psi = \mathbf{X}\psi'$.

[\Rightarrow] Suppose $\sigma^i, N_i^\sigma, \theta \models \mathbf{X}\psi'$. It follows that $\sigma^{i+1}, N_{i+1}^\sigma, \lambda_i^\sigma \circ \theta \models \psi'$. By the induction hypothesis, $(\psi', \Theta) \in D_{i+1}$ such that $\Theta = h_{i+1} \circ [\lambda_i^\sigma \circ \theta]$; moreover, $h_{i+1} \circ \lambda_i^\sigma = \lambda_i \circ h_i$. Since λ_i^σ is injective, it follows that $[\lambda_i^\sigma \circ \theta] = \lambda_i^\sigma \circ [\theta]$ and hence $\Theta = \lambda_i \circ h_i \circ [\theta]$. By the definition of transitions in the tableau graph (Def. 5.11), we may conclude that $(\mathbf{X}\psi', \Theta) \in D_i$.

[\Leftarrow] Suppose $(\mathbf{X}\psi', \Theta) \in D_i$ with $\Theta = h_i \circ [\theta]$. Due to the definition of transitions in the tableau graph, we may conclude $(\psi', \lambda_i \circ \Theta) \in D_{i+1}$. Note that (just as above) $\lambda_i \circ \Theta = h_{i+1} \circ [\lambda_i^\sigma \circ \theta]$ and hence (due to the induction hypothesis) $\sigma^{i+1}, N_{i+1}^\sigma, \lambda_i^\sigma \circ \theta \models \psi'$. It follows that $\sigma^i, N_i^\sigma, \theta \models \mathbf{X}\psi'$.

- **case** $\psi = \psi_1 \cup \psi_2$. For this case we repeatedly use the following correspondence, which holds for all $j \geq i$ (provable by induction on j , using the properties of the generator $(h_i)_{i \in \mathbb{N}}$, see Def. 3.6):

$$\lambda_{j-1} \circ \dots \circ \lambda_i \circ h_i \circ [\theta] = h_j \circ [\lambda_{j-1}^\sigma \circ \dots \circ \lambda_i^\sigma \circ \theta] . \quad (20)$$

[\Rightarrow] Suppose $\sigma^i, N_i^\sigma, \theta \models \psi_1 \cup \psi_2$. It follows that there is a $n \geq i$ such that

1. $\sigma^j, N_j^\sigma, \lambda_{j-1}^\sigma \circ \dots \circ \lambda_i^\sigma \circ \theta \models \psi_1$ for all $i \leq j < n$;
2. $\sigma^n, N_n^\sigma, \lambda_{n-1}^\sigma \circ \dots \circ \lambda_i^\sigma \circ \theta \models \psi_2$.

By the induction hypothesis, and using (20), it follows that

1. $(\psi_1, \lambda_{j-1} \circ \dots \circ \lambda_i \circ h_i \circ [\theta] \upharpoonright \psi_1) \in D_j$ for all $i \leq j < n$;
2. $(\psi_1, \lambda_{n-1} \circ \dots \circ \lambda_i \circ h_i \circ [\theta] \upharpoonright \psi_2) \in D_n$.

But then one can show (using the definition of atom) that also $(\mathbf{X}(\psi_1 \cup \psi_2), \lambda_{j-1} \circ \dots \circ \lambda_i \circ [\theta] \upharpoonright \psi_1) \in D_j$ for all $i \leq j < n$. (This is shown by induction starting at $j = n - 1$ and going down to $j = i$.) We may conclude that, in all cases, $(\psi_1 \cup \psi_2, h_i \circ [\theta]) \in D_i$.

[\Leftarrow] Suppose $(\psi_1 \cup \psi_2, \Theta) \in D_i$ with $\Theta = h_i \circ [\theta]$. By Definition 5.14 (condition 3) there exists an $n \geq i$ such that $(\psi_2, \lambda_{n-1} \circ \dots \circ \lambda_i \circ \Theta \upharpoonright \psi_2) \in D_n$. Let n be the smallest such; then it follows (due to Def. 5.8) that $(\psi_1, \lambda_{j-1} \circ \dots \circ \lambda_i \circ \Theta \upharpoonright \psi_1) \in D_j$ for all $i \leq j < n$. (This is proved by induction on $j \in \{i, \dots, n-1\}$, using the fact that $(\psi_2, \lambda_{j-1} \circ \dots \circ \lambda_i \circ \Theta \upharpoonright \psi_2) \notin D_j$.) Using (20) we get

- $\sigma^j, N_j^\sigma, \lambda_{j-1}^\sigma \circ \dots \circ \lambda_i^\sigma \circ \theta \models \psi_1$ for all $i \leq j < n$;
- $\sigma^n, N_n^\sigma, \lambda_{n-1}^\sigma \circ \dots \circ \lambda_i^\sigma \circ \theta \models \psi_2$.

This implies $\sigma^i, N_i^\sigma, \theta \models \psi_1 \cup \psi_2$. □

Proposition 5.15. A path π in $G_{\mathcal{H}}(\phi)$ fulfills ϕ if and only if there exists $(\phi, \Theta) \in D_0$ (for some Θ) such that $I_{\mathcal{H}}(q_0) = \overline{\Theta}$.

Proof. (**only if**) If π fulfills ϕ there exist a (σ, N, θ) generated from the underlying run ρ by a generator $(h_i)_{i \in \mathbb{N}}$ such that $\sigma, N, \theta \models \phi$ and $k_0 = \min(K(\phi), \Omega(h_0))$. By Lemma D.2, $(\phi, h_0 \circ [\theta]) \in D_0$. Furthermore, we have $h_0 \circ [\theta] = h_0 \circ \theta = I_{\mathcal{H}}(q_0)$.

(**if**) By Lemma D.2, taking the triple (σ, N, θ) generated by $(h_i)_{i \in \mathbb{N}}$ such that $k_0 = \min(K(\phi), \Omega(h_0))$ if $(\phi, \Theta) \in D_0$ with $I_{\mathcal{H}}(q_0) = h_0 \circ \theta = \overline{h_0 \circ [\theta]} = \Theta$ then $\sigma, N, \theta \models \phi$. The existence of the right allocational sequence with the right number of initial imploded entities is explicitly proved. Thus, ϕ is satisfiable. Since (σ, N, θ) is generated by the underlying run of π and the condition of k_0 is satisfied, by definition it follows that π fulfills ϕ . □

Proposition 5.16. ϕ is \mathcal{H} -satisfiable if and only if there exists a path in $G_{\mathcal{H}}(\phi)$ that fulfills ϕ .

Proof. If there exists π in $G_{\mathcal{H}}(\phi)$ that fulfills ϕ , by definition this implies that ϕ is satisfied by an allocation triple (σ, N, θ) generated by the underlying run of π .

Now assume that ϕ is \mathcal{H} -satisfiable, and let $\rho = q_0 \lambda_0 q_1 \lambda_1 \dots$ be a run generating a triple (σ, N, θ) with generator $(h_i)_{i \in \mathbb{N}}$ such that $\sigma, N, \theta \models \phi$. We construct a path $\pi = (q_0, D_0, k_0) \lambda_0 (q_1, D_1, k_1) \lambda_1 \dots$ that fulfills ϕ . For all $i \in \mathbb{N}$ let

$$D_i = \{(\psi, h_i \circ [\theta]) \mid \psi \in CL(\phi), \sigma^i, N_i^\sigma, \theta \models \psi\}$$

and

$$k_i = \min(K(\phi), \Omega(h_i)).$$

It can be proved (by induction on the structure of the formulae in $CL(\phi)$) that the D_i and k_i satisfy the conditions of Definition 5.8; i.e., (q_i, D_i, k_i) is an atom for all $i \in \mathbb{N}$. We show that π is a path by proving that the conditions of Definition 5.14 are satisfied by π . Since π then fulfills ϕ by construction, we are done.

1. By the construction of π .
2. By contradiction. Assume that there exists an $i \geq 0$ such that $(q_i, D_i, k_i) \rightarrow_{\lambda_i} (q_{i+1}, D_{i+1}, k_{i+1})$ is not a transition of G . Take the least such i . Then one of the following must hold:
 - i) $q_i \rightarrow_{\lambda_i} q_{i+1}$ is not a transition in \mathcal{H} . But this contradicts the fact that ρ is a run of \mathcal{H} .
 - ii) there exists $(X\psi, \Theta) \in D_i$ such that $(\psi, \lambda_i \circ \Theta) \notin D_{i+1}$. By the properties of the generator we have that $\lambda_i \circ \Theta = h_{i+1} \circ [\lambda_i^\sigma \circ \theta]$ (see also (20)); hence this would imply $\sigma^{i+1}, N_{i+1}^\sigma, \lambda_i^\sigma \circ \theta \not\models \psi$. But then also $\sigma^i, N_i^\sigma, \theta \not\models X\psi$, contradicting the construction of D_i .
If $(\psi, \lambda_i \circ \Theta) \in D_{i+1}$, but $(X\psi, \Theta) \notin D_i$ then again for the properties of the generator, $\sigma^{i+1}, N_{i+1}^\sigma, \lambda_i^\sigma \circ \theta \models \psi$ and by the semantics of ATL, $\sigma^i, N_i^\sigma, \theta \models X\psi$. Thus by definition of D we must have $(X\psi, \Theta) \in D_i$. Contradiction.
 - iii) $k_{i+1} \neq \min(K(\phi), k_i + \Omega(\lambda_i))$. Since $\Omega(h_{i+1}) = \Omega(h_i) + \Omega(\lambda_i)$, this is also contradictory.

3. Assume $(\psi_1 \cup \psi_2, \Theta) \in D_i$. By the construction of D_i , $\sigma^i, N_i^\sigma, \theta \models \psi_1 \cup \psi_2$ and $\Theta = h_i \circ [\theta]$. Therefore $\sigma^j, N_j^\sigma, \lambda_{j-1}^\sigma \circ \dots \circ \lambda_i^\sigma \circ \theta \upharpoonright fv(\psi_2) \models \psi_2$ for some $j \geq i$. Due to (20) and the construction of D_j , it follows that $(\psi, \lambda_{j-1} \circ \dots \circ \lambda_i \circ \Theta \upharpoonright \psi_2) \in D_j$.

□

Proposition 5.18. If π is fulfilling path in $G_{\mathcal{H}}(\phi)$, then $Inf(\pi)$ is a self-fulfilling SCS of $G_{\mathcal{H}}(\phi)$.

Proof. Let $G' = Inf(\pi)$. G' is strongly connected. From the definition of infinite set it follows that there exists $i \geq 0$ such that the atoms in $\pi^i = (q_i, D_i, k_i)\lambda_i(q_{i+1}, D_{i+1}, k_{i+1})\lambda_{i+1} \dots$ are precisely those in G' . Furthermore, if there is an atom $A \in G'$ such that $(\psi_1 \cup \psi_2, \Theta) \in D_A$, then there exists $j \geq i : (q_j, D_j, k_j) = A$. By condition 3 of Def. 5.14 we have that there exists $n \geq j$ such that $(\psi_2, \lambda_{n-1} \circ \dots \circ \lambda_j \circ \Theta \upharpoonright \psi_2) \in D_n$. But then $(q_n, D_n, k_n) \in G'$, hence G' is self-fulfilling. □

Proposition 5.19. Let $G' \subseteq G_{\mathcal{H}}(\phi)$ be self-fulfilling SCS such that

- there exists a fulfilling prefix of G' starting at an initial atom A with $(\phi, \Theta) \in D_A$ such that $I_{\mathcal{H}}(q_A) = \overline{\Theta}$;
- for all $F \in \mathcal{F}_{\mathcal{H}} : F \cap \{q_B | B \in G'\} \neq \emptyset$;

Then there exists a path π in $G_{\mathcal{H}}(\phi)$ that fulfils ϕ and such that $Inf(\pi) = G'$.

Proof. Satisfaction of an until-valuation. In a (finite or infinite) transition sequence through $G_{\mathcal{H}}(\phi)$, say $A_0 \rightarrow_{\lambda_0} A_1 \rightarrow_{\lambda_1} \dots$, we call an until-valuation $v = (\psi_1 \cup \psi_2, \Theta) \in D_{A_0}$ *satisfied at A_i* (or just *satisfied*) if $(\psi_2, \lambda_{i-1} \circ \dots \circ \lambda_0 \circ \Theta \upharpoonright \psi_2) \in D_{A_i}$.

Observation. Due to the properties of atoms and of transitions in $G_{\mathcal{H}}(\phi)$, if v is not satisfied at any A_j with $j \leq i$, then it follows that $(\psi_1 \cup \psi_2, \lambda_{i-1} \circ \dots \circ \lambda_0 \circ \theta) \in D_{A_i}$.

By the properties assumed for $G_{\mathcal{H}}(\phi)$, there exists a sequence $\pi_1 = A_0 \lambda_0 \dots \lambda_{m-1} A_m$, such that $A = A_0$, $A_i \rightarrow_{\lambda_i} A_{i+1}$ for all $0 \leq i < m$, and $A_m = B$ is a node of G' . Furthermore, starting from B it is possible to construct a finite transition sequence

$$B = B_0 \rightarrow_{\mu_0} B_1 \rightarrow_{\mu_1} \dots \rightarrow_{\mu_{i-1}} B_i$$

in which all $(\psi_1 \cup \psi_2, \Theta) \in D_{B_j}$ are satisfied, in the above sense. The existence of such a transition sequence can be proved by contradiction: suppose that the minimal number of until-valuations $(\psi_1 \cup \psi_2, \Theta) \in D_{B_j}$ that remain unsatisfied in any finite fragment starting at B_0 is u (> 0); take an optimal transition sequence that leaves u until-valuations unsatisfied, and let $(\psi_1 \cup \psi_2, \Theta) \in D_{B_j}$ be one of the unsatisfied ones. As observed above, it follows that $v' = (\psi_1 \cup \psi_2, \mu_{i-1} \circ \dots \circ \mu_0 \circ \Theta) \in D_{B_i}$. However, due to the fact that G' is self-fulfilling, there is a transition sequence

$$B_i \rightarrow_{\mu_i} \dots \rightarrow_{\mu_{n-1}} B_n$$

that satisfies v' , i.e., such that $(\psi_2, \mu_{n-1} \circ \dots \circ \mu_i \circ \dots \circ \mu_0 \circ \Theta \upharpoonright \psi_2) \in D_{B_n}$. But then the combined sequence starting at B_0 and going through B_i to B_n leaves at most $u - 1$ until-valuations of B_0 unsatisfied; contradiction.

We extend this finite transition sequence to a cycle $\pi_2 = B_0 \lambda_0 B_1 \lambda_1 \dots \lambda_{n-1} B_n$, with $B_n = B_0$, that visits all nodes of G' (note that π_2 exists because G' is strongly connected).

Let $\pi = \pi_1 \cdot \pi_2^\omega$. We show that π is an allocational path. Since (by assumption) $(\phi, \Theta) \in D_A$ such that $I_{\mathcal{H}}(q_A) = \overline{\Theta}$ it then follows (by Prop. 5.15) that π fulfils ϕ . Since clearly $Inf(\pi) = G'$ we are then done.

For this purpose we show that the conditions of Def. 5.14 hold.

1. Let $\rho = q_0\lambda_0q_1\lambda_1\cdots$ be the underlying run of π . Note that for all $i < n$ and for all $k \in \mathbb{N}$, $q_{m+i+n*k} = q_{B_i}$. We show that ρ is a run of \mathcal{H} .
 $q_0 \in \text{dom}(I_{\mathcal{H}})$ is guaranteed by assumption on A ; furthermore, $q_i \rightarrow_{\lambda_i} q_{i+1}$ by construction of the graph $G_{\mathcal{H}}(\phi)$. Finally, take an arbitrary $F \in \mathcal{F}_{\mathcal{H}}$. By the assumption in the proposition, we have that $q_B \in F$ for some $B \in G'$; hence $B = B_i$ for some $i < n$. Since then $q_{m+i+k*n} = q_B$ for all $k \in \mathbb{N}$, q_B is visited infinitely often by ρ .
2. By construction of π .
3. Assume $v = (\psi_1 \cup \psi_2, \Theta)$ is in one of the atoms in π . We have to show that v is satisfied somewhere during the sequence. We distinguish three cases:
 - If $v \in D_{A_i}$ for $i < m$ then either $(\psi_2, \lambda_{j-1} \circ \cdots \circ \lambda_i \circ \Theta \upharpoonright \psi_2) \in D_{A_j}$ for some $i \leq j < m$, or $(\psi_1 \cup \psi_2, \lambda_{m-1} \circ \cdots \circ \lambda_i \circ \Theta) \in D_{B_0}$. The latter case is dealt with below.
 - If $v \in D_{B_i}$ for $0 < i < n$, then either $(\psi_2, \lambda_{j-1} \circ \cdots \circ \lambda_i \circ \Theta \upharpoonright \psi_2) \in D_{B_j}$ for some $i \leq j < n$, or $(\psi_1 \cup \psi_2, \lambda_{n-1} \circ \cdots \circ \lambda_i \circ \Theta) \in D_{B_0}$. The latter case is dealt with below.
 - Otherwise, $v \in D_{B_0}$. Due to the construction of π , v is satisfied in one of the B_i ($0 \leq i < n$).

□

Theorem 5.20. For any HABA \mathcal{H} and formula ϕ , it is decidable whether or not ϕ is \mathcal{H} -satisfiable.

Proof. By Propositions 5.15 and 5.16, in order to prove that ϕ is \mathcal{H} -satisfiable, it is necessary and sufficient to search in the graph $G_{\mathcal{H}}(\phi)$ for a fulfilling path π . By Propositions 5.19, it is necessary and sufficient to check only for a self-fulfilling SCS that has a fulfilling prefix whose initial atom (q_0, D_0, k_0) contains (ϕ, Θ) for some Θ such that $\bar{\Theta} = I_{\mathcal{H}}(q_0)$ and has a non-empty intersection with every set of final states. Since SCS are finite objects and there are only a finite number of them, this search can be effective and exhaustive. □