

# Propositional Dynamic Logic for Hyperproperties

Jens Oliver Gutsfeld, Markus Müller-Olm, and Christoph Ohrem

Institut für Informatik, Westfälische Wilhelms-Universität Münster  
Einsteinstraße 62, 48149, Münster, Germany  
{jens.gutsfeld, markus.mueller-olm, christoph.ohrem}@uni-muenster.de

**Abstract.** Information security properties of reactive systems like non-interference often require relating different executions of the system to each other and following them simultaneously. Since common logics like LTL, CTL, or the modal  $\mu$ -calculus cannot express such *hyperproperties*, the hyperlogics HyperLTL and HyperCTL\* were developed to cure this defect. However, these logics are not able to express arbitrary  $\omega$ -regular properties. In this paper, we introduce HyperPDL- $\Delta$ , an adaptation of the Propositional Dynamic Logic of Fischer and Ladner for hyperproperties, in order to remove this limitation. Using an elegant automata-theoretic framework, we show that HyperPDL- $\Delta$  model checking is asymptotically not more expensive than HyperCTL\* model checking, despite its vastly increased expressive power. We further investigate fragments of HyperPDL- $\Delta$  with regard to satisfiability checking.

## 1 Introduction

Temporal logics for model checking like LTL, CTL or CTL\* have been used successfully in verification. The underlying mechanism is to analyse paths of a structure (in linear time logics) or paths and their possible extensions (in branching time logics). Notably, since these logics cannot refer to multiple paths at once, they cannot express *hyperproperties* that relate multiple paths to each other. Examples of hyperproperties include information security properties like non-interference which requires that different executions of a system that receive the same values in their low security variables as inputs behave equally on low security variables. In order to develop a dedicated logic for these properties, Clarkson et. al. [3,10] introduced HyperLTL and HyperCTL\*, which extend LTL and CTL\* by path variables. However, just like LTL and CTL cannot express arbitrary  $\omega$ -regular properties of traces, it has been shown that these logics cannot express all  $\omega$ -regular properties for even singleton sets of traces [19], a desirable property of specification logics [1,14]. Logics like Propositional Dynamic Logic (PDL) [11,16] and Linear Dynamic Logic (LDL) [6,8] are able to do so for single traces. As we seek to extend this ability to hyperproperties, we consider a variant of PDL for hyperproperties called HyperPDL- $\Delta$  in this paper. HyperPDL- $\Delta$  properly extends logics like HyperLTL, HyperCTL\*, LDL and PDL- $\Delta$  and can express all  $\omega$ -regular properties over hypertraces in a handy formalism based

on regular expressions over programs. We develop a model checking algorithm for HyperPDL- $\Delta$  inspired by one for HyperCTL\* [10] and show that the model checking problem for HyperPDL- $\Delta$  is decidable at no higher asymptotic cost than the corresponding problem for HyperCTL\*, in spite of the vastly increased expressive power. Our algorithm non-trivially differs from the algorithm in [10] in two ways: first of all, our algorithm handles more general regular modalities that subsume the modalities of HyperCTL\* and require different constructions. Then, we use a different notion of alternation depth (called *criticality*) which conservatively extends their notion, but requires handling structurally different operators and regular expressions. We also show that for fragments of HyperPDL- $\Delta$  similar to the fragments of HyperLTL in [9], the satisfiability problem is decidable.

This paper is structured as follows: Section 2 introduces Kripke Transition Systems and (Alternating) Büchi Automata. In Section 3, we define our new logic HyperPDL- $\Delta$  and describe properties expressible in it. Afterwards, in Section 4, we outline a model checking algorithm for HyperPDL- $\Delta$  and show that it is asymptotically optimal by providing a precise complexity classification. In Section 5, we consider fragments of HyperPDL- $\Delta$  for which the satisfiability problem is decidable. Then, in Section 6, we show that HyperPDL- $\Delta$  can express all  $\omega$ -regular properties over sets of traces and compare it to existing hyperlogics with regard to expressivity. Finally, in Section 7, we provide a summary of this paper. Due to lack of space, some proofs can be found in the appendix.

**Related Work.** Hyperproperties were systematically analysed in [4] and dedicated temporal logics for hyperproperties, HyperLTL and HyperCTL\*, were introduced in [3]. An overview of temporal hyperlogics and discussion of their expressive power can be found in [5]. Efficient model checking algorithms for these logics were introduced in [10] by Finkbeiner et al. and our model checking algorithm for HyperPDL- $\Delta$  builds on ideas from their construction. Our satisfiability algorithm, on the other hand, is inspired by the corresponding algorithm for HyperLTL [9]. A different line of research for properties involving multiple traces at once is given by *epistemic temporal logics* [12]. An attempt to unify epistemic temporal logics and hyperlogics is given by Bozzelli et. al in [2] via a variant of HyperCTL\* with past modalities. PDL was originally introduced in [11] by Fischer and Ladner and has been extended in multiple ways [16]. There are several attempts to extend temporal logic by regular properties: a variant of PDL for linear time properties of finite traces, LDL<sub>f</sub>, was introduced in [6] and was extended upon for the infinite setting in [8] by introducing parametrised operators. Other regular extensions of temporal logics were studied e.g. by Wolper [21] or by Kupferman et. al [14]. However, all these extensions do not concern hyperproperties.

## 2 Preliminaries

Let  $AP$  be a finite set of atomic propositions and  $\Sigma$  a finite set of atomic programs. A *Kripke Transition System* is a tuple  $\mathcal{T} = (S, s_0, \{\delta_\sigma | \sigma \in \Sigma\}, L)$  where  $S$  is a finite set of states,  $s_0$  is an initial state,  $\delta_\sigma \subseteq S \times S$  is a transition relation

for each  $\sigma \in \Sigma$  and  $L : S \rightarrow 2^{AP}$  is a labeling function. We assume that there are no states without outgoing edges, that is for each  $s \in S$ , there is  $s' \in S$  with  $(s, s') \in \delta_\sigma$  for some  $\sigma \in \Sigma$ . A *KTS* is a combination of a Kripke Structure and a labeled transition system where a Kripke Structure  $K$  is a *KTS* for  $|\Sigma| = 1$  and an *LTS*  $T$  is a *KTS* with the labelling function  $s \mapsto \emptyset$  for all  $s \in S$ . A path in a *KTS*  $\mathcal{T}$  is an infinite alternating sequence  $s_0\sigma_0s_1\sigma_1\ldots \in (S\Sigma)^\omega$  where  $s_0$  is the initial state of  $\mathcal{T}$  and  $(s_i, s_{i+1}) \in \delta_{\sigma_i}$  for all  $i \geq 0$ . We denote by  $Paths(\mathcal{T}, s)$  the set of paths in  $\mathcal{T}$  starting in  $s$  and by  $Paths^*(\mathcal{T}, s)$  the set of corresponding path suffixes  $\{p[i, \infty] \mid p \in Paths(\mathcal{T}, s), i \in \mathbb{N}_0\}$  where  $p[i, \infty]$  is the path suffix of  $p$  starting at index  $i$ . A trace is an alternating infinite sequence  $t \in (2^{AP}\Sigma)^\omega$ . For a path  $\pi = s_0\sigma_0s_1\sigma_1\ldots$ , the induced trace is given by  $L(s_0)\sigma_0L(s_1)\sigma_1\ldots$ . For a *KTS*  $\mathcal{T}$  and a state  $s \in S$ , we write  $Traces(\mathcal{T}, s)$  to denote the traces induced by paths of  $\mathcal{T}$  starting in  $s$ .

An alternating Büchi automaton (ABA) is a tuple  $\mathcal{A} = (Q, q_0, \Sigma, \rho, F)$  where  $Q$  is a finite set of states,  $q_0 \in Q$  is an initial state,  $\Sigma$  is a finite alphabet,  $\rho : Q \times \Sigma \rightarrow \mathbb{B}^+(Q)$  is a transition function mapping each pair of state and input symbol to a non-empty positive boolean combination of successor states and  $F \subseteq Q$  is a set of accepting states. We assume that every ABA has two distinct states  $true \in F$  and  $false \in Q \setminus F$  with  $\rho(true, \sigma) = true$  and  $\rho(false, \sigma) = false$  for all  $\sigma \in \Sigma$ . Thus, all maximal paths in an ABA are infinite. A tree  $T$  is a subset of  $\mathbb{N}^*$  such that for every node  $t \in \mathbb{N}^*$  and every positive integer  $n \in \mathbb{N}$ :  $t \cdot n \in T$  implies (i)  $t \in T$  (we then call  $t \cdot n$  a child of  $t$ ), and (ii) for every  $0 < m < n$ ,  $t \cdot m \in T$ . We assume every node has at least one child. A path in a tree  $T$  is a sequence of nodes  $t_0t_1\ldots$  such that  $t_0 = \varepsilon$  and  $t_{i+1}$  is a child of  $t_i$  for all  $i \in \mathbb{N}_0$ . A run of an ABA  $\mathcal{A}$  on an infinite word  $w \in \Sigma^\omega$  is defined as a  $Q$ -labeled tree  $(T, r)$  where  $r : T \rightarrow Q$  is a labelling function such that  $r(\varepsilon) = q_0$  and for every node  $t \in T$  with children  $t_1, \dots, t_k$ , we have  $1 \leq k \leq |Q|$  and the valuation assigning true to the states  $r(t_1), \dots, r(t_k)$  and false to all other states satisfies  $\rho(r(t), w(|t|))$ . A run  $(T, r)$  is an accepting run iff for every path  $t_1t_2\ldots$  in  $T$ , there are infinitely many  $i$  with  $r(t_i) \in F$ . A word  $w$  is accepted by  $\mathcal{A}$  iff there is an accepting run of  $\mathcal{A}$  on  $w$ . The set of infinite words accepted by  $\mathcal{A}$  is denoted by  $\mathcal{L}(\mathcal{A})$ . A nondeterministic Büchi automaton is an ABA in which every transition rule consists only of disjunctions.

We will make use of two well-known theorems about ABA:

**Proposition 1** ([17]). *For every ABA  $\mathcal{A}$  with  $n$  states, there is a nondeterministic Büchi automaton  $MH(\mathcal{A})$  with  $2^{\mathcal{O}(n)}$  states that accepts the same language.*

**Proposition 2** ([17],[15]). *For every ABA  $\mathcal{A}$  with  $n$  states, there is an ABA  $\overline{\mathcal{A}}$  with  $\mathcal{O}(n^2)$  states that accepts the complement language, i.e.,  $\mathcal{L}(\overline{\mathcal{A}}) = \overline{\mathcal{L}(\mathcal{A})}$ .*

### 3 Propositional Dynamic Logic for Hyperproperties

In this section, we define our new logic, HyperPDL- $\Delta$ . We use the syntax of HyperCTL\* as a basis and replace the modalities  $\bigcirc, \mathcal{U}$  and  $\mathcal{R}$  by PDL-like expressions  $\langle \alpha \rangle \varphi, [\alpha] \varphi$  and  $\Delta \alpha$ , where  $\alpha$  is a regular expression over tuples of

atomic programs  $\tau$  describing behaviour on quantified paths. Additionally, we allow  $\tau$ -Operators in  $\alpha$  to enable constructions like conditional branching.

**Definition 1 (Syntax of HyperPDL- $\Delta$ ).** *Let  $N = \{\epsilon, \pi_1, \pi_2 \dots\}$  be a set of path variables with a special path variable  $\epsilon \in N$ . A formula  $\varphi$  is a HyperPDL- $\Delta$  formula if it is built from the following context-free grammar:*

$$\begin{aligned}\varphi &::= \exists \pi. \varphi \mid \forall \pi. \varphi \mid a_\pi \mid \neg \varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle \alpha \rangle \varphi \mid [\alpha] \varphi \mid \Delta \alpha \\ \alpha &::= \tau \mid \epsilon \mid \alpha + \alpha \mid \alpha \alpha \mid \alpha^* \mid \varphi?\end{aligned}$$

where  $\pi \in N \setminus \{\epsilon\}$  and  $\tau \in (\Sigma \cup \{\cdot\})^n$  for the number  $n > 0$  of path quantifiers that  $\alpha$  is in scope of. Constructs  $\langle \alpha \rangle \varphi$ ,  $[\alpha] \varphi$  and  $\Delta \alpha$  are only allowed in scope of at least one quantifier. Terms constructed from  $\alpha$  are called programs.

We call a HyperPDL- $\Delta$  formula  $\varphi$  *closed* iff all occurrences of path variables  $\pi$  in  $\varphi$  are bound by a quantifier. In this paper, we only consider closed formulas  $\varphi$ . In programs  $\alpha$ , each component of tuples  $\tau \in (\Sigma \cup \{\cdot\})^n$  corresponds to one of the path variables bound by a quantifier  $\alpha$  is in scope of.

Connectives inherited from HyperCTL\* are interpreted analogously: Connectives  $\exists$  and  $\forall$  should be read as “along some path” and “along all paths”. Using different path variables  $\pi$  enables us to refer to multiple paths at the same time. For example, with  $\forall \pi_1. \exists \pi_2. \exists \pi_3. \varphi$ , one can express that for all paths  $\pi_1$ , there are paths  $\pi_2$  and  $\pi_3$  such that  $\varphi$  holds along these three paths. Boolean connectives are defined in the usual way. Atomic propositions  $a \in AP$  express information about a state and have to be indexed by a path variable  $\pi$  to express on which path we expect  $a$  to hold.

Expressions  $\alpha$  should be interpreted as regular programs on path prefixes with the ability to check properties  $\varphi$  of infinite suffixes along the way. Just as formulas, these regular programs refer to multiple paths. The path prefixes to be considered end at the same index and atomic programs on all paths have to be taken into account. In formulas  $\varphi$ , one can compare atomic propositions on different paths using the syntax  $a_\pi$  since boolean connectives are available. Referring to occurrences of atomic programs on single paths in a similar way would reduce expressivity since neither negation nor conjunction are present in  $\alpha$ . Instead, we use tuples  $\tau \in (\Sigma \cup \{\cdot\})^n$  to refer to atomic programs on paths  $\pi_1, \dots, \pi_n$ , where  $\sigma$  in position  $i$  means that on path  $\pi_i$ , we expect a use of  $\sigma$  to reach the next state. A wildcard symbol  $\cdot$  expresses that any atomic program is allowed on the corresponding path.

Intuitively, the formula constructs using  $\alpha$  can be interpreted as follows:  $\langle \alpha \rangle \varphi$  means that there is a prefix of the current paths matching  $\alpha$  after which  $\varphi$  holds.  $[\alpha] \varphi$  is the dual of  $\langle \alpha \rangle \varphi$ , meaning  $\varphi$  holds at the end of all prefixes matching  $\alpha$ . The last construct  $\Delta \alpha$  is of a different kind and expresses  $\omega$ -regular rather than regular properties. It says that  $\alpha$  should occur repeatedly, i.e. the currently quantified paths can be divided into infinitely many segments matching  $\alpha$ .  $\Delta \alpha$  expresses a variant of a Büchi condition. Instead of moving from accepting states to accepting states in a Büchi automaton, one moves from initial states to accepting states repeatedly.

Using our logic, one can easily and intuitively express common hyperproperties. Consider the following two examples: the first hyperproperty, *observational determinism* [4], which can be expressed by  $\forall \pi_1. \forall \pi_2. (\bigwedge_{a \in L} a_{\pi_1} \leftrightarrow a_{\pi_2}) \rightarrow [\bullet^*] \bigwedge_{a \in L} a_{\pi_1} \leftrightarrow a_{\pi_2}$ , states that if two executions of a system receive equal low security inputs, they are indistinguishable for a low security observer all the time, where low security observable behaviour is modelled by the atomic propositions in  $L$ . Here, we use the common boolean abbreviations  $\rightarrow$  for implication and  $\leftrightarrow$  for equivalence as well as  $\bullet$  as an abbreviation the program  $\tau = (\cdot, \dots, \cdot)$ . The second hyperproperty, *generalized noninterference* [4], states that high security injections do not interfere with low security observable behaviour. It can be expressed by stating that for all pairs of executions  $\pi_1, \pi_2$  there is a third execution  $\pi_3$  agreeing with  $\pi_1$  on high security injections and is indistinguishable from  $\pi_2$  for a low security observer:  $\forall \pi_1. \forall \pi_2. \exists \pi_3. [\bullet^*] \bigwedge_{a \in H} a_{\pi_1} \leftrightarrow a_{\pi_3} \wedge \bigwedge_{a \in L} a_{\pi_2} \leftrightarrow a_{\pi_3}$ . Since these hyperproperties can already be expressed in HyperLTL [3], which is subsumed by our logic by encoding  $\varphi_1 \mathcal{U} \varphi_2$  formulas with  $\langle (\varphi_1 ? \bullet)^* \rangle \varphi_2$ , it is no surprise, that they can be expressed in HyperPDL- $\Delta$  as well.

The ability to express arbitrary  $\omega$ -regular properties however, allows a much more fine-grained analysis of a system than HyperLTL. For example, by replacing the program  $\bullet^*$  with  $(\bullet\bullet)^* \bullet (\sigma_1, \sigma_1)$  in the observational determinism formula, we can restrict the requirement on low security outputs to only apply for every other state with the additional constraint that some specific program  $\sigma_1$  was last executed in both  $\pi_1$  and  $\pi_2$ . This is particularly helpful in approaches like the Assume Guarantee Paradigm [18], where in order to establish *guarantees*  $\varphi_G$  about a system, certain *assumptions*  $\varphi_A$  about the system can be used. In order to establish  $\mathcal{T} \models \varphi_G$ , one can then instead check  $\mathcal{T} \models \varphi_A \rightarrow \varphi_G$ , enabling verification of more powerful guarantees  $\varphi_G$  about  $\mathcal{T}$ . For example, if one is convinced that certain loops  $\text{while}(a)\{\sigma\}$  are irrelevant for security considerations, those executions can be excluded from  $\pi_1$  by adding the conjunct  $[(a_{\pi_1} ? (\sigma, \cdot))^* \neg a_{\pi_1} ?] \text{false}$  to  $\varphi_A$ . Such fine-grained reasoning about the system behaviour is particularly necessary for hardware systems according to Armoni et. al. [1].

While there are hyperlogics with the ability to express all  $\omega$ -regular languages [5], these lack properties desirable for verification: HyperQPTL obtains the ability to express  $\omega$ -regular properties from the addition of propositional quantification, which makes it hard to use for specification purposes since the user has to keep track of heterogeneous types of quantifiers. The simple property that all executions  $\pi_1$  and  $\pi_2$  agree on propositions from a set  $P$  in every other state for example is encoded by the HyperQPTL formula  $\forall \pi_1. \forall \pi_2. \exists t : t \wedge \Box(\bigcirc t \leftrightarrow \neg t) \wedge \Box(t \rightarrow \bigwedge_{a \in P} a_{\pi_1} \leftrightarrow a_{\pi_2})$  [13]. The specification of this property in HyperPDL- $\Delta$  is much more direct:  $\forall \pi_1. \forall \pi_2. [(\bullet\bullet)^*] \bigwedge_{a \in P} a_{\pi_1} \leftrightarrow a_{\pi_2}$ . This example also illustrates another problem: due to the additional quantifier alternation, the only known model checking algorithm for HyperQPTL [19] is exponentially more expensive than that of HyperPDL- $\Delta$  for such formulas. S1S[E] on the other hand, while being even more expressive than HyperPDL- $\Delta$ , has an undecidable model checking problem [5].

Before formally defining our logic's semantics, we introduce some notation. We call a partial function  $\Pi : N \rightsquigarrow Paths^*(\mathcal{T}, s_0)$  a path assignment and denote by  $PA$  the set of all path assignments. While  $N$  can be infinite, we are only interested in the subset  $dom(\Pi)$  of  $N$  when dealing with path assignments  $\Pi$ . In the context of a formula  $\varphi$ , these are exactly the bound variables and  $\epsilon$ , which makes  $dom(\Pi)$  finite. We use  $\{\}$  to denote the empty path assignment with  $dom(\{\}) = \emptyset$ . We introduce  $\Pi[i, \infty]$  as a notation to manipulate path assignments  $\Pi$  such that  $\Pi[i, \infty](\pi) = \Pi(\pi)[i, \infty]$  holds for all  $\pi \in dom(\Pi)$ . Also,  $\Pi[\pi_i \rightarrow p]$  is a notation for a path assignment  $\Pi'$  where  $\Pi'(\pi_i) = p$  and  $\Pi'(\pi_j) = \Pi(\pi_j)$  for all  $j \neq i$ . Furthermore, we use  $\epsilon$  to refer to the most recently assigned path in a path assignment. As a convention, we do not count  $\epsilon$  when determining  $|dom(\Pi)|$ .

**Definition 2 (Semantics of HyperPDL- $\Delta$ ).** *Given a KTS  $\mathcal{T}$ , we inductively define both satisfaction of formulas  $\varphi$  and programs  $\alpha$  on path assignments  $\Pi$ .*

$$\begin{aligned}
\Pi \models_{\mathcal{T}} \exists \pi. \varphi & \quad \text{iff there is } p \in Paths(\mathcal{T}, \Pi(\epsilon)(0)) \text{ s.t. } \Pi[\pi \rightarrow p, \epsilon \rightarrow p] \models_{\mathcal{T}} \varphi \\
\Pi \models_{\mathcal{T}} \forall \pi. \varphi & \quad \text{iff for all } p \in Paths(\mathcal{T}, \Pi(\epsilon)(0)) : \Pi[\pi \rightarrow p, \epsilon \rightarrow p] \models_{\mathcal{T}} \varphi \\
\Pi \models_{\mathcal{T}} a_{\pi} & \quad \text{iff } a \in L(\Pi(\pi)(0)) \\
\Pi \models_{\mathcal{T}} \neg \varphi & \quad \text{iff } \Pi \not\models_{\mathcal{T}} \varphi \\
\Pi \models_{\mathcal{T}} \varphi_1 \wedge \varphi_2 & \quad \text{iff } \Pi \models_{\mathcal{T}} \varphi_1 \text{ and } \Pi \models_{\mathcal{T}} \varphi_2 \\
\Pi \models_{\mathcal{T}} \varphi_1 \vee \varphi_2 & \quad \text{iff } \Pi \models_{\mathcal{T}} \varphi_1 \text{ or } \Pi \models_{\mathcal{T}} \varphi_2 \\
\Pi \models_{\mathcal{T}} \langle \alpha \rangle \varphi & \quad \text{iff there is } i \geq 0 \text{ s.t. } \Pi[i, \infty] \models_{\mathcal{T}} \varphi \text{ and } (\Pi, 0, i) \in R(\alpha) \\
\Pi \models_{\mathcal{T}} [\alpha] \varphi & \quad \text{iff for all } i \geq 0 \text{ with } (\Pi, 0, i) \in R(\alpha) : \Pi[i, \infty] \models_{\mathcal{T}} \varphi \\
\Pi \models_{\mathcal{T}} \Delta \alpha & \quad \text{iff there are } 0 = k_1 \leq k_2 \leq \dots \text{ s.t. for all } i \geq 1 : \\
& \quad (\Pi, k_i, k_{i+1}) \in R(\alpha)
\end{aligned}$$

$$\begin{aligned}
(\Pi, i, k) \in R(\tau) & \quad \text{iff } k = i + 2 \text{ and for all } 1 \leq l \leq dom(\Pi) : \\
& \quad \tau|_l = \cdot \text{ or } \Pi(\pi_l)(i + 1) = \tau|_l \\
(\Pi, i, k) \in R(\varepsilon) & \quad \text{iff } i = k \\
(\Pi, i, k) \in R(\alpha_1 + \alpha_2) & \quad \text{iff } (\Pi, i, k) \in R(\alpha_1) \text{ or } (\Pi, i, k) \in R(\alpha_2) \\
(\Pi, i, k) \in R(\alpha_1 \alpha_2) & \quad \text{iff there is } j \text{ s.t. } i \leq j \leq k, (\Pi, i, j) \in R(\alpha_1) \text{ and} \\
& \quad (\Pi, j, k) \in R(\alpha_2) \\
(\Pi, i, k) \in R(\alpha^*) & \quad \text{iff there are } l \geq 0, i = j_0 \leq j_1 \leq \dots \leq j_l = k \text{ s.t.} \\
& \quad \text{for all } 0 \leq m < l : (\Pi, j_m, j_{m+1}) \in R(\alpha) \\
(\Pi, i, k) \in R(\varphi?) & \quad \text{iff } i = k \text{ and } \Pi[i, \infty] \models_{\mathcal{T}} \varphi
\end{aligned}$$

A KTS  $\mathcal{T} = (S, s_0, \{\delta_{\sigma} | \sigma \in \Sigma\}, L)$  satisfies a formula  $\varphi$ , denoted by  $\mathcal{T} \models \varphi$ , iff  $\{\} \models_{\mathcal{T}} \varphi$  holds for the empty path assignment  $\{\}$ . For a well-defined semantics, we define  $\{\}(\epsilon)(0)$ , which is needed to determine whether  $\{\} \models_{\mathcal{T}} Q\pi.\varphi$  for  $Q \in \{\exists, \forall\}$  holds, to yield the initial state  $s_0$  of  $\mathcal{T}$ . Note that  $(\Pi, i, k) \in R(\alpha)$  can hold for even  $i$  and  $k$  only.

## 4 Model Checking HyperPDL- $\Delta$

In order to tackle the model checking problem for HyperPDL- $\Delta$ , that is to check whether  $\mathcal{T} \models \varphi$  holds for arbitrary *KTSs*  $\mathcal{T}$  and HyperPDL- $\Delta$  formulas  $\varphi$ , we develop a new model checking algorithm inspired by the HyperCTL\* model checking algorithm from [10]. We assume a naming convention where a formula  $\varphi$  is normalised such that a quantifier that is in scope of  $i - 1$  other quantifiers quantifies path variable  $\pi_i$ . This can easily be achieved by bounded renaming of path variables without changing the semantics. By this convention, only path assignments  $\Pi$  with  $\text{dom}(\Pi) = \{\pi_1, \dots, \pi_n, \epsilon\}$  appear for a subformula that is in the scope of  $n$  quantifiers. The main idea of the model checking algorithm is to represent such a path assignment  $\Pi$  by an  $\omega$ -word over  $(S^n \times \Sigma^n)$ . Formally, we define a translation function  $\nu : PA \rightarrow \bigcup_{n \in \mathbb{N}_0} (S^n \times \Sigma^n)^\omega$  such that a path assignment  $\Pi$  with  $\Pi(\pi_i) = s_i^0 \sigma_i^0 s_i^1 \sigma_i^1 \dots$  is mapped to  $\nu(\Pi) = ((s_0^0, \dots, s_n^0), (\sigma_0^0, \dots, \sigma_n^0))((s_0^1, \dots, s_n^1), (\sigma_0^1, \dots, \sigma_n^1)) \dots \in (S^n \times \Sigma^n)^\omega$  for  $n = |\text{dom}(\Pi)|$ . Note that  $\epsilon$  does not have to be encoded separately in  $\nu(\Pi)$  since  $\Pi(\pi_n) = \Pi(\epsilon)$  always holds. We use the notation  $s$  and  $\tau$  to refer to tuples  $(s_1, \dots, s_n)$  and  $(\sigma_1, \dots, \sigma_n)$  and write  $s|_i$  or  $\tau|_i$  to refer to  $s_i$  and  $\sigma_i$ , respectively. Then, given a formula  $\varphi$  and a *KTS*  $\mathcal{T}$ , we construct an ABA  $\mathcal{A}_\varphi$  recognising  $\nu(\Pi)$  iff  $\Pi \models_{\mathcal{T}} \varphi$  holds.

In our construction, we first transform all formulas into a variation of negation normal form, where only existential quantifiers are allowed and negation can only occur in front of existential quantifiers, atomic propositions or  $\Delta$ s. The main difference from conventional NNF is that universal quantifiers are not allowed here. This is due to the fact, that negated existential quantifiers can be handled in a better way than universal quantifiers in our setup. The transformation is done by driving all negations in the formula inwards using De Morgan's laws and the duality  $\neg \langle \alpha \rangle \varphi \equiv [\alpha] \neg \varphi$  while also replacing universal quantifiers  $\forall$  with  $\neg \exists \neg$  and cancelling double negations successively. For example, the formula  $\forall \pi. [\alpha] \neg a_\pi$  is transformed into  $\neg \exists \pi. \neg [\alpha] \neg a_\pi$ ,  $\neg \exists \pi. \langle \alpha \rangle \neg \neg a_\pi$  and then  $\neg \exists \pi. \langle \alpha \rangle a_\pi$  successively.

Then, as another preprocessing step, all programs  $\alpha$  appearing in the formula are inductively translated to an intermediate automaton representation  $M_\alpha = (Q, q_0, \Sigma^n, \rho, q_f, \Psi)$ . Automata  $M_\alpha$  can be seen as nondeterministic finite automata (NFA) with access to oracles which, started from an index  $i$ , recognise all prefixes up to index  $j$  of  $\nu(\Pi[i, \infty])$  such that  $(\Pi, i, j) \in R(\alpha)$ . This is formalised in Lemma 1. The only differences in syntax when compared to a conventional NFA are (i) there is exactly one final state  $q_f$  instead of a set  $F$ , (ii)  $\varepsilon$ -edges are not eliminated and (iii) we have a state marking function  $\Psi$  mapping every state  $q \in Q$  to a singleton or empty set of formulas  $\Psi(q)$ . These state markings  $\Psi(q)$  are introduced to tackle tests  $\psi?$  and are later replaced by transitions to automata  $\mathcal{A}_\psi$ , which we define in our main construction. These state markings make the standard elimination of  $\varepsilon$ -edges impossible, which is why we delay the elimination until the markings are eliminated as well. This approach is similar to the one used in [8] to transform LDL formulas into automata. However, we apply the construction in a hyperlogic context and make it more succinct by offering

$$\begin{array}{ll}
\tau & M_\alpha = (\{q_0, q_1\}, q_0, \Sigma^n, \rho, q_1, \Psi), \Psi(q_i) = \emptyset \\
& \rho(q, (\sigma_1, \dots, \sigma_n)) = \begin{cases} \{q_1\} & \text{if } \forall i. \tau|_i = \cdot \vee \tau|_i = \sigma_i \text{ and } q = q_0 \\ \emptyset & \text{else} \end{cases} \\
& \rho(q, \varepsilon) = \emptyset \\
\varepsilon & M_\alpha = (\{q_0, q_1\}, q_0, \Sigma^n, \rho, q_1, \Psi), \Psi(q_i) = \emptyset \\
& \rho(q_i, \tau) = \emptyset \\
& \rho(q_i, \varepsilon) = \begin{cases} \{q_{i+1}\} & \text{if } i = 0 \\ \emptyset & \text{else} \end{cases} \\
\alpha_1 + \alpha_2 & M_\alpha = (Q_1 \dot{\cup} Q_2 \dot{\cup} \{q_0, q_f\}, q_0, \Sigma^n, \rho, q_f, \Psi), \\
& \Psi(q_0) = \Psi(q_f) = \emptyset, \Psi(q) = \Psi_i(q) \text{ for } q \in Q_i \\
& \rho(q, \tau) = \begin{cases} \rho_i(q, \tau) & \text{if } q \in Q_i \\ \emptyset & \text{else} \end{cases} \\
& \rho(q, \varepsilon) = \begin{cases} \{q_{0,1}, q_{0,2}\} & \text{if } q = q_0 \\ \rho_i(q, \varepsilon) & \text{if } q \in Q_i \setminus \{q_{f,i}\} \\ \rho_i(q, \varepsilon) \cup \{q_f\} & \text{if } q = q_{f,i} \end{cases} \\
\alpha_1 \alpha_2 & M_\alpha = (Q_1 \dot{\cup} Q_2, q_{0,1}, \Sigma^n, \rho, q_{f,2}, \Psi), \Psi(q) = \Psi_i(q) \text{ for } q \in Q_i \\
& \rho(q, \tau) = \rho_i(q, \tau) \text{ for } q \in Q_i \\
& \rho(q, \varepsilon) = \begin{cases} \rho_i(q, \varepsilon) & \text{if } q \in Q_i, q \neq \{q_{f,1}\} \\ \rho_1(q, \varepsilon) \cup \{q_{0,2}\} & \text{if } q = q_{f,1} \end{cases} \\
(\alpha_1)^* & M_\alpha = (Q_1, q_{0,1}, \Sigma^n, \rho, q_{0,f}, \Psi), \Psi(q) = \Psi_1(q), \text{ for } q \in Q_1 \\
& \rho(q, \tau) = \rho_1(q, \tau) \\
& \rho(q, \varepsilon) = \begin{cases} \rho_1(q, \varepsilon) & \text{if } q \notin \{q_{0,1}, q_{0,f}\} \\ \rho_1(q, \varepsilon) \cup \{q_{0,f}\} & \text{if } q = q_{0,1} \\ \rho_1(q, \varepsilon) \cup \{q_{0,1}\} & \text{if } q = q_{0,f} \end{cases} \\
\psi? & M_\alpha = (\{q_0, q_1, q_2\}, q_0, \Sigma^n, \rho, q_2, \Psi), \Psi(q_0) = \Psi(q_2) = \emptyset, \Psi(q_1) = \{\psi\} \\
& \rho(q, \tau) = \emptyset \\
& \rho(q_i, \varepsilon) = \begin{cases} \{q_{i+1}\} & \text{if } i = 0, 1 \\ \emptyset & \text{else} \end{cases}
\end{array}$$

**Fig. 1.** Construction of  $M_\alpha$

an alternative approach to constructing the transition function, thus avoiding an exponential blowup, and eliminating unnecessary states.

**Construction of  $M_\alpha$ :** We now describe the construction of  $M_\alpha$ . When an expression  $\alpha$  has one or more subexpressions  $\alpha_i$ , we assume that automata  $M_{\alpha_i} = (Q_i, q_{0,i}, \Sigma^n, \rho_i, q_{f,i}, \Psi_i)$  are already constructed. Constructions can be seen in Figure 1.

Let  $\stackrel{\varepsilon}{\Rightarrow}_X \subseteq Q \times Q$  for a set of formulas  $X$  be the smallest relation such that (i)  $q \stackrel{\varepsilon}{\Rightarrow}_{\Psi(q)} q$  and (ii)  $q' \stackrel{\varepsilon}{\Rightarrow}_X q''$  and  $q' \in \rho(q, \varepsilon)$  imply  $q \stackrel{\varepsilon}{\Rightarrow}_{X \cup \Psi(q)} q''$ . Then for  $\tau \in \Sigma^n$ ,  $\stackrel{\tau}{\Rightarrow}_X$  is the smallest relation such that  $q \stackrel{\tau}{\Rightarrow}_X q''$  iff there is a  $q' \in Q$  such that  $q \stackrel{\varepsilon}{\Rightarrow}_X q'$  and  $q'' \in \rho(q', \tau)$ . This relation captures the encountered markings along  $\varepsilon$ -paths in  $M_\alpha$ . By working with a relation on states instead of with the  $\varepsilon$ -paths directly as in [8], we alleviate the need to isolate starting and final states of the automata, which keeps the construction more succinct.

We obtain the following Lemma:



$$\begin{aligned}
a_{\pi_k} \quad \mathcal{A}_\varphi &= (\{q_0\}, q_0, \Sigma_\varphi, \rho, \emptyset) \\
\rho(q_0, (s, \tau)) &= \begin{cases} \text{true} & \text{if } a \in L(s|_k) \\ \text{false} & \text{else} \end{cases} \\
\neg a_{\pi_k} \quad \mathcal{A}_\varphi &= (\{q_0\}, q_0, \Sigma_\varphi, \rho, \emptyset) \\
\rho(q_0, (s, \tau)) &= \begin{cases} \text{false} & \text{if } a \in L(s|_k) \\ \text{true} & \text{else} \end{cases} \\
\varphi_1 \wedge \varphi_2 \quad \mathcal{A}_\varphi &= (Q_1 \dot{\cup} Q_2 \dot{\cup} \{q_0\}, q_0, \Sigma_\varphi, \rho, F_1 \dot{\cup} F_2) \\
\rho(q, (s, \tau)) &= \begin{cases} \rho_1(q_{0,1}, (s, \tau)) \wedge \rho_2(q_{0,2}, (s, \tau)) & \text{if } q = q_0 \\ \rho_i(q, (s, \tau)) & \text{if } q \in Q_i, i \in \{1, 2\} \end{cases} \\
\varphi_1 \vee \varphi_2 \quad \mathcal{A}_\varphi &= (Q_1 \dot{\cup} Q_2 \dot{\cup} \{q_0\}, q_0, \Sigma_\varphi, \rho, F_1 \dot{\cup} F_2) \\
\rho(q, (s, \tau)) &= \begin{cases} \rho_1(q_{0,1}, (s, \tau)) \vee \rho_2(q_{0,2}, (s, \tau)) & \text{if } q = q_0 \\ \rho_i(q, (s, \tau)) & \text{if } q \in Q_i, i \in \{1, 2\} \end{cases}
\end{aligned}$$

**Fig. 2.** Construction of  $\mathcal{A}_\varphi$  in easy cases

**Lemma 1.** *Let  $\Pi$  be a path assignment with  $\nu(\Pi) = (s_0\tau_1)(s_2\tau_3)\dots$  and  $\alpha$  a program, then  $(\Pi, i, k) \in R(\alpha)$  iff there is a state sequence  $q_0q_1\dots q_m$  with  $m = \frac{k-i}{2}$  in  $M_\alpha$  and sets of formulas  $X_0, \dots, X_m$  such that*

- (i)  $q_0$  is the initial state of  $M_\alpha$ ,
- (ii)  $q_m \xrightarrow{\varepsilon}_{X_m} q_f$  for the final state  $q_f$  of  $M_\alpha$ ,
- (iii)  $q_l \xrightarrow{\tau_l + 2l + 1}_{X_l} q_{l+1}$  for all  $l < m$  and
- (iv)  $\psi \in X_l$  implies  $\Pi[i + 2l, \infty] \models_{\mathcal{T}} \psi$  for all  $l \leq m$ .

As mentioned, we transform  $\varphi$  into an ABA  $\mathcal{A}_\varphi$  recognising  $\nu(\Pi)$  iff  $\Pi \models_{\mathcal{T}} \varphi$  holds. Formally, a language  $\mathcal{L} \subseteq (S^n \times \Sigma^n)^\omega$  is called  $\mathcal{T}$ -equivalent to a formula  $\varphi$ , if for each  $\Pi$  the statements  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_\varphi)$  and  $\Pi \models_{\mathcal{T}} \varphi$  are equivalent. Broadening the definition, we say that an ABA  $\mathcal{A}$  is  $\mathcal{T}$ -equivalent to  $\varphi$  iff its language  $\mathcal{L}(\mathcal{A})$  is  $\mathcal{T}$ -equivalent to  $\varphi$ . As a closed formula  $\varphi$  is a boolean combination of quantified subformulas  $\psi$ , the model checking problem can be reduced to nonemptiness checks on  $\mathcal{A}_\psi$  for all maximal quantified subformulas  $\psi$ .

**Construction of  $\mathcal{A}_\varphi$ :** We construct the ABA  $\mathcal{A}_\varphi$  inductively. The alphabet of  $\mathcal{A}_\varphi$  is given as  $\Sigma_\varphi = S^n \times \Sigma^n$  where  $n$  is the number of path quantifiers the formula  $\varphi$  is in scope of. When constructing an automaton for  $\varphi$  with subformulas  $\varphi_i$ , we assume that the automata  $\mathcal{A}_{\varphi_i} = (Q_{\varphi_i}, q_{0,\varphi_i}, \Sigma_{\varphi_i}, \rho_{\varphi_i}, F_{\varphi_i})$  are already constructed. Similarly, when a formula contains an expression  $\alpha$ , we assume that not only  $M_\alpha = (Q_\alpha, q_{0,\alpha}, \Sigma^n, \rho_\alpha, q_{f,\alpha}, \Psi)$  but also  $\mathcal{A}_{\psi_i}$  in the case of  $\langle \alpha \rangle \varphi$  and  $\Delta \alpha$  or  $\mathcal{A}_{\psi'_i}$  in case of  $[\alpha] \varphi$  for each construct  $\psi_i$  in  $\alpha$  are already constructed. Here,  $\psi'_i$  is the negation normal form of  $\neg \psi_i$ . Recall that states *true* and *false* are always part of an ABA in our definition, so we will not mention them explicitly. Furthermore, let  $\mathcal{T} = (S, s_0, \{\delta_\sigma | \sigma \in \Sigma\}, L)$  be the *KTS* to be checked.

In the easy cases of the construction seen in Figure 2, it is mostly straightforward to see how the automata work. For  $a_{\pi_k}$  (resp.  $\neg a_{\pi_k}$ ), only those words are accepted where the state first read for path  $\pi_k$  is labelled with  $a$  (resp. not labelled with  $a$ ). The connectives  $\wedge$  and  $\vee$  are implemented into our ABA straightforwardly: given  $\mathcal{A}_{\varphi_1}$  and  $\mathcal{A}_{\varphi_2}$  one checks whether both automata (resp. at least one automaton) accept the word.

$$\begin{aligned}
\langle \alpha \rangle \varphi_1 \mathcal{A}_\varphi &= (Q_1 \dot{\cup} Q_\alpha \dot{\cup} \bigcup_i Q_{\psi_i}, q_{0,\alpha}, \Sigma_\varphi, \rho, F_1 \dot{\cup} \bigcup_i F_{\psi_i}) \\
\rho(q, (s, \tau)) &= \begin{cases} \rho_1(q, (s, \tau)) & \text{if } q \in Q_1 \\ \rho_{\psi_i}(q, (s, \tau)) & \text{if } q \in Q_{\psi_i} \\ \bigvee \{q' \wedge \bigwedge_{\psi_i \in X} \rho_{\psi_i}(q_{0,\psi_i}, (s, \tau)) | q \xrightarrow{\tau}_X q'\} \cup \\ \quad \{\rho(q_{0,1}, (s, \tau)) \wedge \\ \quad \bigwedge_{\psi_i \in X} \rho_{\psi_i}(q_{0,\psi_i}, (s, \tau)) | q \xrightarrow{\varepsilon}_X q_{f,\alpha}\} & \text{if } q \in Q_\alpha \end{cases} \\
[\alpha] \varphi_1 \mathcal{A}_\varphi &= (Q_1 \dot{\cup} Q_\alpha \dot{\cup} \bigcup_i Q_{\neg\psi_i}, q_{0,\alpha}, \Sigma_\varphi, \rho, F_1 \dot{\cup} Q_\alpha \dot{\cup} \bigcup_i F_{\neg\psi_i}) \\
\rho(q, (s, \tau)) &= \begin{cases} \rho_1(q, (s, \tau)) & \text{if } q \in Q_1 \\ \rho_{\neg\psi_i}(q, (s, \tau)) & \text{if } q \in Q_{\neg\psi_i} \\ \bigwedge \{q' \vee \bigvee_{\psi_i \in X} \rho_{\neg\psi_i}(q_{0,\neg\psi_i}, (s, \tau)) | q \xrightarrow{\tau}_X q'\} \cup \\ \quad \{\rho(q_{0,1}, (s, \tau)) \vee \\ \quad \bigvee_{\psi_i \in X} \rho_{\neg\psi_i}(q_{0,\neg\psi_i}, (s, \tau)) | q \xrightarrow{\varepsilon}_X q_{f,\alpha}\} & \text{if } q \in Q_\alpha \end{cases} \\
\Delta\alpha \mathcal{A}_\varphi &= (Q_\alpha \dot{\cup} \{q_0\} \dot{\cup} \bigcup_i Q_{\psi_i}, q_0, \Sigma_\varphi, \rho, \{q_0\} \dot{\cup} \bigcup_i F_{\psi_i}) \\
\rho(q, (s, \tau)) &= \begin{cases} \rho_{\psi_i}(q, (s, \tau)) & \text{if } q \in Q_{\psi_i} \\ \bigvee \{q' \wedge \bigwedge_{\psi_i \in X} \rho_{\psi_i}(q_{0,\psi_i}, (s, \tau)) | q_{0,\alpha} \xrightarrow{\tau}_X q'\} \cup \\ \quad \{\bigwedge_{\psi_i \in X} \rho_{\psi_i}(q_{0,\psi_i}, (s, \tau)) | q_{0,\alpha} \xrightarrow{\varepsilon}_X q_{f,\alpha}\} & \text{if } q = q_0 \\ \bigvee \{q' \wedge \bigwedge_{\psi_i \in X} \rho_{\psi_i}(q_{0,\psi_i}, (s, \tau)) | q \xrightarrow{\tau}_X q'\} \cup \\ \quad \{q_0 \wedge \bigwedge_{\psi_i \in X} \rho_{\psi_i}(q_{0,\psi_i}, (s, \tau)) \wedge \\ \quad \bigwedge_{\psi_i \in Y} q_{0,\psi_i} | q \xrightarrow{\tau}_X q' \xrightarrow{\varepsilon}_Y q_{f,\alpha}\} & \text{if } q \in Q_\alpha \end{cases} \\
\neg\Delta\alpha \mathcal{A}_\varphi &= \overline{\mathcal{A}_{\Delta\alpha}}
\end{aligned}$$

**Fig. 3.** Construction of  $\mathcal{A}_\varphi$  for  $\alpha$  formulas

As seen in the previous construction, we use an intermediate representation  $M_\alpha$  to check whether a prefix of a given path assignment satisfies  $\alpha$ . In Lemma 1 (iv) we make use of oracle requests to prove that our construction works when  $\alpha$  contains tests  $\psi?$ . As stated in  $M_\alpha$ 's construction, we eliminate these oracle requests in the construction of  $\mathcal{A}_\varphi$  by transitioning into the automaton  $\mathcal{A}_\psi$  whenever we reach a state marked with  $\psi?$ .

In our  $\langle \alpha \rangle \varphi_1$  construction, we want to recognise a single path where after a prefix satisfying  $\alpha$ ,  $\varphi_1$  holds. This is done by giving the possibility to move into  $\mathcal{A}_{\varphi_1}$  whenever a final state of  $M_\alpha$  is reached. Since we keep none of the final states of  $M_\alpha$  as final states in  $\mathcal{A}_\varphi$ , an accepting run in  $\mathcal{A}_\varphi$  finally has to move into  $\mathcal{A}_{\varphi_1}$  via a state in  $F_\alpha$  and thus cannot stay in  $M_\alpha$  forever. For the dual case  $[\alpha] \varphi_1$ , we want  $\varphi_1$  to hold after all prefixes satisfying  $\alpha$ . Thus, dual to the previous construction, whenever reaching a state in  $F_\alpha$ , we are not only given the possibility to, but have to move into  $\mathcal{A}_{\varphi_1}$  as well. Since all transitions in this construction are combined with  $\wedge$  to ensure that all prefixes satisfying  $\alpha$  are considered, we have to take care of paths in  $\mathcal{A}_\varphi$  that never leave  $M_\alpha$  by making all states of  $M_\alpha$  accepting. On the other hand, paths not satisfying  $\alpha$  due to a violation of a test  $\psi?$  are ruled out by a disjunctive test for the negation of  $\psi$ . To handle formulas of the form  $\Delta\alpha$ , we transform  $M_\alpha$  into a Büchi automaton with the acceptance condition that in between two visits of a final state,  $\alpha$  is satisfied. We move to the start of the automaton whenever a final state was reached. We thus introduce an initial state  $q_0$  acting as an initial state  $q_{0,\alpha}$  for

outgoing, and as a final state  $q_{f,\alpha}$  for incoming transitions. We make this state the only accepting state (apart from those in the  $\mathcal{A}_{\psi_i}$  automata) to ensure that repeated visits of final states are only possible with a “reset” of the automaton.

Note that when we translate state markings in  $M_\alpha$  into transitions in  $\mathcal{A}_\varphi$ , there can be situations where  $q \xrightarrow{X} q'$  holds for exponentially many  $X$ , resulting in a transition function of exponential size in  $|\alpha|$ . This can however be avoided by taking an alternative approach and inductively constructing a transition function equivalent to  $\bigvee \{ \bigwedge X | q \xrightarrow{X} q' \}$  for  $\langle \cdot \rangle$  and  $\Delta$  or equivalent to  $\bigwedge \{ \bigvee X | q \xrightarrow{X} q' \}$  for  $[\cdot]$  during the inductive construction of  $M_\alpha$ . The number of paths with different behaviour introduced during the construction for  $\alpha^*$  can then be armortised against the size of  $\alpha$ , resulting in a transition function not larger than  $2 \cdot |\alpha|$ . We refer the interested reader to the appendix of this paper for a more detailed look at this alternative construction.

$$\begin{aligned}
\exists \pi. \varphi_1 \quad \mathcal{A}_{\varphi_1} \text{ dealternised: } MH(\mathcal{A}_{\varphi_1}) &= (Q_1, q_{0,1}, \Sigma_{\varphi_1}, \rho_1, F_1) \\
\mathcal{A}_\varphi &= (Q_1 \times S \times \Sigma \cup \{q_0\}, q_0, \Sigma_\varphi, \rho, F_1 \times S \times \Sigma) \\
\rho(q_0, (s, \tau)) &= \{(q', s', \sigma') | q' \in \rho_1(q_{0,1}, s + s|_n, \tau + \sigma), s' \in \delta_\sigma(s|_n), \sigma, \sigma' \in \Sigma\} \\
\rho((q, s, \sigma), (s, \tau)) &= \{(q', s', \sigma') | q' \in \rho_1(q, s + s, \tau + \sigma), s' \in \delta_\sigma(s), \sigma' \in \Sigma\} \\
\neg \exists \pi. \varphi_1 \quad \mathcal{A}_\varphi &= \mathcal{A}_{\exists \pi. \varphi_1}
\end{aligned}$$

**Fig. 4.** Construction of  $\mathcal{A}_\varphi$  for quantifier formulas

In the constructions for path quantifiers (Figure 4), we remove one component of the alphabet  $\Sigma_{\varphi_1} = S^{n+1} \times \Sigma^{n+1}$  and switch to  $\Sigma_\varphi = S^n \times \Sigma^n$ . The missing component is now simulated by the state space of  $\mathcal{A}_\varphi$ , also making sure that said component is indeed a path in  $\mathcal{T}$ . Intuitively,  $\mathcal{A}_\varphi$  simulates paths of  $\mathcal{T}$  starting in different states and accepts those satisfying  $\varphi_1$ . Our approach for the transformation of a formula to negation normal form implies that we also have to consider negated existential quantifiers, which are handled by a straightforward complementation construction.

For the construction, we obtain the following theorem via induction:

**Theorem 1.** *The automaton  $\mathcal{A}_\varphi$  is  $\mathcal{T}$ -equivalent to  $\varphi$ .*

For the complexity analysis, we need a notion of criticality.

**Definition 3 (Criticality).** *A HyperPDL- $\Delta$  formula  $\varphi$  in negation normal form has criticality equal to the highest number of critical quantifiers along any of the paths of the formula’s syntax tree. Any quantifier is called critical iff it is a non-outermost quantifier, fulfills at least one of the following three conditions:*

- (i) *it is a negated quantifier,*
  - (ii) *it is an outermost quantifier in a test  $\varphi?$  in some program  $\alpha$ ,*
  - (iii) *it is an outermost quantifier in the subformula  $\varphi$  of  $[\alpha]\varphi$ ,*
- and is not a negated outermost quantifier in a test  $\varphi?$  occurring in a modality  $[\alpha]$  where the automaton  $M_\alpha$  is deterministic.*

*We call a HyperPDL- $\Delta$  formula with criticality 0 uncritical.*

This definition is designed carefully in order to ensure that the alternation depth of HyperCTL\* formulas coincides with the criticality of their direct translation to HyperPDL- $\Delta$ .

Since the size of the constructed automaton is in the worst case increased exponentially whenever we use the dealternation construction  $MH(\mathcal{A})$ , we can only bound the size by an exponential tower of height  $k$  which we define as  $g_{p,c}(0, n) = p(n)$  and  $g_{p,c}(k + 1, n) = c^{g(k, n)}$  for  $c > 1$  and a polynomial  $p$ . We use  $\mathcal{O}(g(k, n))$  as an abbreviation for  $\mathcal{O}(g_{p,c}(k, n))$  for some  $c > 1$  and polynomial  $p$ . An exponential blowup can however be avoided, when *uncritical* constructions are performed in between two dealternation constructions. Thus, the height of the tower is determined by criticality rather than quantifier depth. Two remarks are in order about the next Lemma. Firstly, the statement refers to the number of states of the construction, disregarding the number of transitions. However, this is harmless for our complexity analysis: while the alphabet size increases exponentially with the nesting depth of quantifiers, alphabets need not be represented explicitly and the number of transitions of the automata can be kept polynomial by delaying the substitution of  $\cdot$  by concrete programs until the intersection with the system  $\mathcal{T}$ . We refrain from explicating this in our construction in order to increase readability. Secondly, the construction's size can also increase exponentially in the nesting level of negated  $\Delta\alpha$  constructions. Since negated  $\Delta\alpha$  constructions are rarely nested in formulas, we assume for the remainder of this paper a bound on this nesting level in order to simplify our complexity statement. In Section 6, we will show that a bound of 1 suffices to express  $\omega$ -regular properties making this a reasonable constraint. The dependency from the nesting depth is reflected in the proof of Lemma 2 in the appendix, which we formulate in the following way:

**Lemma 2.** *For a formula  $\varphi$  with criticality  $k$ , the automaton  $\mathcal{A}_\varphi$  has size  $\mathcal{O}(g(k + 1, |\varphi|) \cdot g(k, |\mathcal{T}|))$ .*

*Proof.* (Sketch) By induction on the criticality  $k$ . In the base case, the dealternation construction from Proposition 1 causes an exponential blowup at most once for the innermost quantifier since in all constructions not increasing the criticality, one has to keep track of only a single state of each dealternised subautomaton in further dealternations. Since the only dealternation is done before the system  $\mathcal{T}$  is folded around the automaton, the automaton's size is only exponential in the size of  $\varphi$ , but not in the size of  $\mathcal{T}$ .

In the inductive step, the construction for the outermost critical quantifier increases the size of the automaton exponentially. For all further constructions, it can be argued that the automaton's size is asymptotically not further increased, just like in the base case.  $\square$

**Theorem 2.** *Given a KTS  $\mathcal{T}$  and a HyperPDL- $\Delta$  formula  $\varphi$  with criticality  $k$ , we can decide whether  $\mathcal{T} \models \varphi$  in  $NSPACE(g(k, |\varphi|) \cdot g(k - 1, |\mathcal{T}|))$ .*

*Proof.* The formula arising from the transformation of  $\varphi$  to our variant of negation normal form is a boolean combination of subformulas  $\psi$  with an outermost

existential quantifier. Due to dealternation for the existential quantifier, the automata  $\mathcal{A}_\psi$  are non-deterministic Büchi automata. It is well-known that the nonemptiness check for Büchi automata is in NLOGSPACE in the size of the automaton [7]. The boolean combination of the results does not add to this complexity. Thus, by Lemma 2, we obtain an  $\text{NSPACE}(g(k, |\varphi|) \cdot g(k-1, |\mathcal{T}|))$  model checking algorithm for criticality  $k$  HyperPDL- $\Delta$  formulas.  $\square$

Since we can easily translate HyperCTL\* formulae to HyperPDL- $\Delta$  while preserving the alternation depth as criticality, we can use known hardness results for HyperCTL\* model checking [19] to obtain the following Theorem:

**Theorem 3.** *Given a KTS  $\mathcal{T}$  and a HyperPDL- $\Delta$  formula  $\varphi$  with criticality  $k$ , the model checking Problem for HyperPDL- $\Delta$  is hard for  $\text{NSPACE}(g(k, |\varphi|))$  and  $\text{NSPACE}(g(k-1, |\mathcal{T}|))$ .*

## 5 Satisfiability

While model checking of temporal logics alone is an essential technique for verification, it requires meaningful specifications in order to be useful. For example, if a formula is always true or always false, it is useless for specification purposes. To evaluate whether this is the case, satisfiability testing can be employed as a sanity check [20]. Since satisfiability checking is already undecidable for HyperLTL [9] via a reduction from the Post Correspondence Problem (PCP) and HyperLTL can be embedded into HyperPDL- $\Delta$ , we consider only restrictions of a linear fragment of HyperPDL- $\Delta$  for the purpose of this section.

**Definition 4.** *A linear HyperPDL- $\Delta$  formula is of the form  $Q_1\pi_1 \dots Q_n\pi_n\psi$  for  $Q_i \in \{\exists, \forall\}$  and  $\psi$  a quantifier-free formula. A linear HyperPDL- $\Delta$  formula  $\varphi$  is an  $\forall^*$  formula if  $Q_i = \forall$  for all  $1 \leq i \leq n$ . The  $\exists^*$ -fragment and the  $\forall^*\exists^*$ -fragment are defined analogously.*

For linear HyperPDL- $\Delta$ , we generalise the semantics to arbitrary sets of traces  $T$  instead of only those induced by KTSs. More concretely, for a fixed set of traces  $T$ , we let the assignment functions  $\Pi$  map variables to traces in  $T$  instead of paths of a structure. Such a function  $\Pi$  is called a trace assignment; the set of all trace assignments is called  $TA$ . Furthermore, we let all quantifiers range over  $T$  instead of  $\text{Paths}(\mathcal{T}, \Pi(\varepsilon)(0))$ . We further define  $T \models \varphi$  to hold for a linear HyperPDL- $\Delta$  formula  $\varphi$  iff  $\Pi \models \varphi$  holds for the empty trace assignment  $\Pi$  over  $T$ . Using  $T = \text{Traces}(\mathcal{T}, s_0)$ , this definition coincides with the definition in Section 3 for linear HyperPDL- $\Delta$  formulas. We then call a linear HyperPDL- $\Delta$  formula  $\varphi$  *satisfiable* iff there is a non-empty set of traces  $T$  such that  $T \models \varphi$  holds. The linear HyperPDL- $\Delta$  satisfiability problem is to check whether a linear HyperPDL- $\Delta$  formula  $\varphi$  is satisfiable. For full linear HyperPDL- $\Delta$ , we again obtain undecidability via a reduction from PCP [9]:

**Theorem 4.** *The satisfiability problem for linear HyperPDL- $\Delta$  is undecidable.*

However, since the encoding relies on the availability of arbitrary combinations of quantifiers, the question is whether fragments of linear HyperPDL- $\Delta$  with restricted quantifier combinations yield a decidable satisfiability problem. For HyperLTL, several such restrictions were considered [9]. In [9], the satisfiability problem for the restricted fragments in HyperLTL is solved via the transformation of a HyperLTL formula to an equisatisfiable LTL formula and solving the satisfiability problem for the latter. Since there is no apparent connection of this form between HyperPDL- $\Delta$  and LTL, we use a similar type of translation, but instead translate our formulas into suitable ABA and check those for emptiness. In all cases, the lower complexity bound can be obtained via reduction from the satisfiability problem of the corresponding fragment of HyperLTL.

**Theorem 5.** *The satisfiability problem for the  $\forall^*$ -fragment of HyperPDL- $\Delta$  is PSPACE-complete.*

*Proof.* (Sketch) Let  $\varphi$  be a  $\forall^*$ -fragment formula, i.e.  $\varphi \equiv \forall\pi_1 \dots \forall\pi_n \psi$  for a quantifier-free formula  $\psi$ . Let  $\psi'$  be the formula obtained from  $\psi$  by replacing every occurrence of an atomic proposition  $a_{\pi_i}$  by the atomic proposition  $a_\pi$  for a fresh path variable  $\pi$ . For example, for  $\varphi \equiv \forall\pi_1 \forall\pi_2 \langle \bullet \rangle a_{\pi_1} \wedge \neg a_{\pi_2}$ ,  $\psi$  is given by  $\langle \bullet \rangle a_{\pi_1} \wedge \neg a_{\pi_2}$  and  $\psi'$  is then given by  $\langle \bullet \rangle a_\pi \wedge \neg a_\pi$ . Let  $\mathcal{A}$  be the ABA for  $\psi'$  as described in Section 4. We can test  $\mathcal{A}$  for emptiness in PSPACE [7].  $\mathcal{A}$  is non-empty iff  $\varphi$  is satisfiable: any word  $w$  accepted by  $\mathcal{A}$  gives rise to the trace set  $\{w\}$  satisfying  $\varphi$ . Analogously, a non-empty trace set  $T$  with  $T \models \varphi$  can be used to obtain a word accepted by  $\mathcal{A}$  since  $\mathcal{A}$  accepts all traces satisfying  $\psi'$  and we can thus pick any trace  $t$  in  $T$  as a witness.  $\square$

Similarly, we can show that the  $\exists^*$  fragment is also PSPACE-complete.

**Theorem 6.** *The satisfiability problem for the  $\exists^*$ -fragment of HyperPDL- $\Delta$  is PSPACE-complete.*

*Proof.* (Sketch) Let  $\varphi$  be a  $\exists^*$ -fragment formula, i.e.  $\varphi \equiv \exists\pi_1 \dots \exists\pi_n \psi$  for a quantifier-free formula  $\psi$ . We construct the alternating Büchi automaton  $\mathcal{A}$  for  $\psi$ . Non-emptiness of  $\mathcal{A}$  and satisfiability of  $\varphi$  are equivalent: every trace set  $T$  fulfilling  $\varphi$  contains traces  $t_1 \dots t_n$  fulfilling  $\psi$  and these give rise to a word accepted by  $\mathcal{A}$ . On the other hand, if  $\mathcal{L}(\mathcal{A})$  is non-empty, the trace set  $T$  induced by an arbitrary  $w \in \mathcal{L}(\mathcal{A})$  fulfills  $\varphi$ .  $\square$

For the  $\exists^*\forall^*$ -fragment, we eliminate the universal quantifiers by taking the variables bound by existential quantifiers and replacing the variables bound by universal quantifiers by all possible combinations of them. Unlike the previous two fragments, this increases the complexity beyond PSPACE.

**Theorem 7.** *The satisfiability problem for the  $\exists^*\forall^*$ -fragment of HyperPDL- $\Delta$  is EXPSpace-complete.*

*Proof.* (Sketch) Let  $\varphi$  be a  $\exists^*\forall^*$ -formula, i.e.  $\varphi \equiv \exists\pi_1 \dots \exists\pi_n \forall\pi'_1 \dots \forall\pi'_m \psi$  for a quantifier-free formula  $\psi$ . For a formula  $\psi$  and path variables  $\pi, \pi'$ , we define the

substitution  $\psi[\pi/\pi']$  to be the variant of  $\psi$  in which all occurrences of  $\pi'$  have been replaced by  $\pi$ . Let  $\varphi' \equiv \exists\pi_1 \dots \exists\pi_n \bigwedge_{j_1=1}^n \dots \bigwedge_{j_m=1}^n \psi[\pi_{j_1}/\pi'_1] \dots [\pi_{j_m}/\pi'_m]$ . For example, for  $\varphi \equiv \exists\pi_1 \exists\pi_2 \forall\pi'_1 \langle \bullet \rangle a_{\pi_1} \wedge \neg a_{\pi_2} \wedge a_{\pi'_1}$ ,  $\varphi' = \exists\pi_1 \exists\pi_2 (\langle \bullet \rangle a_{\pi_1} \wedge \neg a_{\pi_2} \wedge a_{\pi_1}) \wedge (\langle \bullet \rangle a_{\pi_1} \wedge \neg a_{\pi_2} \wedge a_{\pi_2})$ .  $\varphi'$  is an  $\exists^*$  formula and is equisatisfiable to  $\varphi$ : any trace assignment satisfying  $\varphi$  naturally induces a model of  $\varphi'$ . For the reverse direction, assume  $\Pi \models \varphi$ .  $\varphi'$  contains all possible combinations of assignments for the variables  $\pi'_1 \dots \pi'_m$  with traces chosen for the existentially quantified variables  $\pi_1 \dots \pi_n$ . Then  $T = \{\Pi(\pi_i) \mid 1 \leq i \leq n\} \models \varphi$ .  $\varphi'$  is constructible in EXPTIME and the satisfiability check is possible in EXPSPACE due to Theorem 6.  $\square$

As in [9], from Theorem 7, we obtain that it is an EXPSPACE-complete problem to decide whether one uncritical HyperPDL- $\Delta$  formula is implied by another. In particular, the uncritical fragment includes properties like variants of observational determinism enriched by regular predicates.

## 6 Expressivity Results

As mentioned in the introduction, a desirable property of temporal logics is the ability to specify arbitrary  $\omega$ -regular properties. We show that HyperPDL- $\Delta$  indeed has this property.

**Theorem 8.** *Let  $\Pi$  be a trace assignment and  $\pi_1 \dots \pi_n$  be the variables bound by  $\Pi$ . Let  $\nu_{AP} : TA \rightarrow ((2^{AP})^n)^\omega$  be the analog of  $\nu$  for trace assignments. For a given  $\omega$ -regular language  $\mathcal{L}$  over  $(2^{AP})^n$ , there is a quantifier-free HyperPDL- $\Delta$  formula  $\varphi$  with path variables  $\pi_1 \dots \pi_n$  such that  $\Pi \models \varphi$  iff  $\nu_{AP}(\Pi) \in \mathcal{L}$ .*

*Proof.* Let  $\mathcal{L}$  be an  $\omega$ -regular language over  $(2^{AP})^n$ . It is well-known [7] that  $\mathcal{L} = \bigcup_{i=1}^k \mathcal{L}_{i,0} \mathcal{L}_{i,1}^\omega$  holds for some regular languages  $\mathcal{L}_{i,0}, \mathcal{L}_{i,1}$ . Let  $r_{i,j}$  be a regular expression for  $\mathcal{L}_{i,j}$ . Every symbol in  $r_{i,j}$  has the form  $(P_1, \dots, P_n)$  for  $P_k \subseteq AP$ . Let  $\alpha_{i,j}$  be the regular expression obtained by replacing each such symbol in  $r_{i,j}$  by  $(\bigwedge_{l=1}^n \bigwedge_{a \in P_l} a_{\pi_l} \wedge \bigwedge_{a \notin P_l} \neg a_{\pi_l})? \bullet$ . Then  $\varphi \equiv \bigvee_{i=1}^k \langle \alpha_{i,0} \rangle \Delta \alpha_{i,1}$  yields the desired formula.  $\square$

We now compare our logic to other hyperlogics. For this purpose we introduce a logic that adds to HyperCTL\* the ability to quantify over atomic propositions.

**Definition 5 ([5]).** *The logic HyperQCTL\* is obtained by adding to the syntax of HyperCTL\* the rules  $\varphi ::= q \mid \exists q. \varphi$  and to the semantics the rules*

$$\begin{aligned} \Pi \models_{\mathcal{T}} q & \quad \text{iff} \quad q \in \Pi(\pi_q)(0) \\ \Pi \models_{\mathcal{T}} \exists q. \varphi & \quad \text{iff} \quad \exists t \in (2^{\{q\}})^\omega. \Pi[\pi_q \rightarrow t] \models_{\mathcal{T}} \varphi \end{aligned}$$

*The sub-logic HyperQPTL consists of the HyperQCTL\* formulas of the form  $Q_1 \pi_1 \dots Q_n \pi_n. q_1 a_1 \dots q_m a_m. \varphi$ , where the  $Q_i$  are path quantifiers, the  $q_j$  are propositional quantifiers, and  $\varphi$  is quantifier-free.*

**Theorem 9.** *1. HyperCTL\* < HyperPDL- $\Delta$   $\leq$  HyperQCTL\**

## 2. Linear HyperPDL- $\Delta$ = HyperQPTL

*Proof.* Part one of the first claim is straightforward: Embedding HyperCTL\* into HyperPDL- $\Delta$  works as described in Section 3. An embedding in the other direction is impossible due to the inability of HyperCTL\* to express arbitrary  $\omega$ -regular properties [19]. For the second part, we observe that the semantics of HyperPDL- $\Delta$  can straightforwardly be encoded in MSO[E], which is equally expressive as HyperQCTL\* by [5].

The  $\subseteq$  direction of the second claim can be shown by a direct translation via Büchi automata: A quantifier-free HyperPDL- $\Delta$  formula can be translated into a Büchi Automaton as described in Section 4. By [13], there is a QPTL formula for that automaton, where the quantifiers can be reapplied to yield the desired formula. For the other direction, consider a HyperQPTL formula  $Q_1\pi_1 \dots Q_n\pi_n.q_1a_1 \dots q_ma_m.\varphi$ . The main idea is to compute a Büchi automaton for  $q_1a_1 \dots q_ma_m.\varphi$ . For this, we first eliminate  $\forall$  quantifiers by replacing them with  $\neg\exists\neg$ . In a second step, we build a Büchi automaton  $\mathcal{A}_\varphi$  for the innermost formula  $\varphi$  in which all  $a_j$  are treated as additional path variables and the tests for  $a_j$  in  $\varphi$  are treated like atomic propositions  $p_{a_j}$  for some atomic proposition  $p$ . Afterwards, we treat quantified propositions  $\exists a_j$  and negations inductively: If the innermost construct is a quantifier  $\exists a_j$ , we eliminate  $a_j$  from the input alphabet by removing the entry for  $a_j$  from every transition, i.e. a value for  $a_j$  will be guessed implicitly. If the innermost construct is a negation, we complement the automaton.<sup>1</sup> The resulting automaton can then be transformed into an  $\omega$ -regular expression before translating that into a HyperPDL- $\Delta$  formula  $\varphi'$  via Theorem 8. Then,  $Q_1\pi_1 \dots Q_n\pi_n.\varphi'$  is the desired HyperPDL- $\Delta$  formula.  $\square$

Due to previous work this places linear HyperPDL- $\Delta$  above HyperLTL and FO[ $\langle, E$ ] in the linear-time hierarchy of hyperlogics [5]. From the same hierarchy, we obtain that linear HyperPDL- $\Delta$  is strictly less expressive than S1S[E], which, as mentioned, has an undecidable model checking problem. We leave a more precise localisation of HyperPDL- $\Delta$  in the branching-time hierarchy of hyperlogics from [5] for future work. If HyperPDL- $\Delta$  turns out to be strictly less expressive than HyperQCTL\*, it would be placed above or incomparable to MPL[E], as the latter logic, unlike HyperPDL- $\Delta$ , cannot express arbitrary  $\omega$ -regular properties.

## 7 Conclusion

We introduced the logic HyperPDL- $\Delta$  as a variant of Propositional Dynamic Logic for hyperproperties that can express all  $\omega$ -regular properties. We indicated that our logic can include fine-grained information in verification while maintaining an intuitive syntax. Our model checking algorithm has the same complexity as model checking HyperCTL\*, despite the increased expressive power. Finally, we showed that satisfiability checking for certain fragments has the same complexity as structurally similar, but less expressive fragments of HyperLTL.

<sup>1</sup> Note that the iterated complementation constructions cause an exponential blowup of the automaton for each propositional quantifier alternation.



## References

1. ARMONI, R., FIX, L., FLAISHER, A., GERTH, R., GINSBURG, B., KANZA, T., LANDVER, A., MADOR-HAIM, S., SINGERMAN, E., TIEMEYER, A., ET AL. The forspec temporal logic: A new temporal property-specification language. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* (2002), Springer, pp. 296–311.
2. BOZZELLI, L., MAUBERT, B., AND PINCHINAT, S. Unifying hyper and epistemic temporal logics. In *FoSSaCS* (2015), vol. 9034 of *Lecture Notes in Computer Science*, Springer, pp. 167–182.
3. CLARKSON, M. R., FINKBEINER, B., KOLEINI, M., MICINSKI, K. K., RABE, M. N., AND SÁNCHEZ, C. Temporal logics for hyperproperties. In *POST 2014* (2014), pp. 265–284.
4. CLARKSON, M. R., AND SCHNEIDER, F. B. Hyperproperties. *Journal of Computer Security* 18, 6 (2010), 1157–1210.
5. COENEN, N., FINKBEINER, B., HAHN, C., AND HOFMANN, J. The hierarchy of hyperlogics. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019* (2019), pp. 1–13.
6. DE GIACOMO, G., AND VARDI, M. Y. Linear temporal logic and linear dynamic logic on finite traces. In *Twenty-Third International Joint Conference on Artificial Intelligence* (2013).
7. DEMRI, S., GORANKO, V., AND LANGE, M. *Temporal Logics in Computer Science: Finite-State Systems*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2016.
8. FAYMONVILLE, P., AND ZIMMERMANN, M. Parametric linear dynamic logic. *Information and Computation* 253 (2017), 237 – 256. GandALF 2014.
9. FINKBEINER, B., AND HAHN, C. Deciding hyperproperties. In *CONCUR 2016* (2016), pp. 13:1–13:14.
10. FINKBEINER, B., RABE, M. N., AND SÁNCHEZ, C. Algorithms for model checking hyperl<sub>tl</sub> and hyperctl<sup>\*</sup>. In *CAV 2015* (2015), pp. 30–48.
11. FISCHER, M. J., AND LADNER, R. E. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.* 18, 2 (1979), 194–211.
12. HALPERN, J. Y., VAN DER MEYDEN, R., AND VARDI, M. Y. Complete axiomatizations for reasoning about knowledge and time. *SIAM J. Comput.* 33, 3 (2004), 674–703.
13. KESTEN, Y., AND PNUELI, A. Complete proof system for QPTL. *J. Log. Comput.* 12, 5 (2002), 701–745.
14. KUPFERMAN, O., PITERMAN, N., AND VARDI, M. Y. Extended temporal logic revisited. In *CONCUR 2001* (2001), pp. 519–535.
15. KUPFERMAN, O., AND VARDI, M. Y. Weak alternating automata are not that weak. *ACM Trans. Comput. Logic* 2, 3 (July 2001), 408–429.
16. LANGE, M. Model checking propositional dynamic logic with all extras. *J. Applied Logic* 4, 1 (2006), 39–49.
17. MIYANO, S., AND HAYASHI, T. Alternating finite automata on omega-words. *Theoretical Computer Science* 32, 3 (1984), 321 – 330.
18. PNUELI, A. In transition from global to modular temporal reasoning about programs. In *Logics and models of concurrent systems*. Springer, 1985, pp. 123–144.
19. RABE, M. N. *A temporal logic approach to Information-flow control*. PhD thesis, Saarland University, 2016.

20. ROZIER, K. Y., AND VARDI, M. Y. LTL satisfiability checking. In *SPIN 2007* (2007), pp. 149–167.
21. WOLPER, P. Temporal logic can be more expressive. *Information and Control* 56, 1/2 (1983), 72–99.

## A Missing proofs from Section 4

*Proof (Lemma 1).* By structural induction over the structure of  $\alpha$ .

The fact that  $i$  and  $k$  are even ensures that the path assignment  $\Pi[i, k]$  starts and ends with states rather than atomic programs.

$\alpha = \tau$ : Assume  $(\Pi, i, k) \in R(\alpha)$ . By definition, this implies  $\tau|_l = \cdot$  or  $\tau_{i+1}|_l = \tau|_l$  for all  $l$  (\*) and  $k = i + 2$ . We construct a state sequence in  $M_\alpha$  satisfying (i) to (iv). Such a state sequence can only consist of two states and only a single symbol can be read due to the length restriction. The state sequence  $q_0 q_3$  with empty sets  $X_0, X_1$  satisfies this constraint as well as (i) and (ii) by construction, (iii) by (\*) and (iv) trivially.

Let there be a state sequence in  $M_\alpha$  satisfying conditions (i) to (iv). By construction and conditions (i) to (iii), we get that the state sequence can only consist of  $q_0$  and  $q_3$ , thus  $k = i + 2$  and  $\tau_{i+1}$  is the only symbol read. Additionally, we get that for all  $l$   $\tau|_l = \cdot$  or  $\tau|_l = \tau_{i+1}|_l$  which establishes  $(\Pi, i, k) \in R(\alpha)$  by definition.

$\alpha = \varepsilon$ : Assume, that  $(\Pi, i, k) \in R(\alpha)$  holds. This implies, by definition, that  $k = i$  and the state sequence we are looking for can only consist of a single state. Indeed, the singleton state sequence  $q_0$  with an empty set  $X_0$  satisfies conditions (i) to (iv): it is the initial state and the final state is  $\varepsilon$ -reachable which establishes (i) and (ii), whereas the remaining conditions (iii) and (iv) are established trivially.

We assume there is a state sequence in  $M_\alpha$  satisfying (i) to (iv). Since there are only  $\varepsilon$ -transitions in  $M_\alpha$ , it can only consist of a single state, the initial state  $q_0$ . Therefore, since  $m = 0$ , we have  $i = k$ , which by definition establishes  $(\Pi, i, k) \in R(\alpha)$ .

$\alpha = \alpha_1 + \alpha_2$ : First, assume that  $(\Pi, i, k) \in R(\alpha)$  holds. By definition, this implies either  $(\Pi, i, k) \in R(\alpha_1)$  or  $(\Pi, i, k) \in R(\alpha_2)$ . We differentiate two cases where the first (resp. the second) condition holds and consider  $(\Pi, i, k) \in R(\alpha_1)$ . The other case is analogous. By induction hypothesis, we get a state sequence  $q_{0,\alpha_1} \dots q_{m,\alpha_1}$  with  $m = \frac{k-i}{2}$  and sets  $X_0, \dots, X_m$  satisfying conditions (i) to (iv). We replace  $q_{0,\alpha_1}$  by  $q_0$  and  $q_{m,\alpha_1}$  by  $q_f$  to obtain a state sequence of the same length in  $M_\alpha$ . Condition (i) is already established.  $\varepsilon$ -transitions from  $q_0$  to  $q_{0,\alpha_1}$  and from  $q_{m,\alpha_1}$  to  $q_f$  make sure that (ii) and (iii) still hold. Finally, since  $q_{0,\alpha_1} \dots q_{m,\alpha_1}$  satisfied (iv) and neither  $q_0$  nor  $q_f$  are marked, (iv) is established as well with the same sets  $X_i$ .

Now, assume there is a state sequence with sets  $X_0, \dots, X_m$  in  $M_\alpha$  satisfying (i) to (iv). Since, by construction,  $M_{\alpha_1}$  and  $M_{\alpha_2}$  are unconnected subautomata of  $M_\alpha$ , the state sequence has to move from  $q_0$  to  $q_f$  either through a state sequence in  $M_{\alpha_1}$  or in  $M_{\alpha_2}$ . We differentiate the two cases and assume the first one. The second case is analogous. We replace  $q_0$  with  $q_{0,\alpha_1}$  and  $q_f$  with  $q_{f,\alpha_1}$  in the full sequence. Since states in  $M_{\alpha_1}$  are only reachable from  $q_0$  via  $q_{0,\alpha_1}$  and  $q_f$  is only reachable from states in  $M_{\alpha_1}$  via  $q_{f,\alpha_1}$ , conditions (i) to (iii) still hold for this state sequence in  $M_{\alpha_1}$ . Condition (iv) is established with the same sets  $X_i$  by the property of our construction that initial and final states can never be

marked. By induction hypothesis, we get that  $(\Pi, i, k) \in R(\alpha_1)$  which satisfies the requirements for  $(\Pi, i, k) \in R(\alpha)$ .

$\alpha = \alpha_1 \alpha_2$ : Assume that  $(\Pi, i, k) \in R(\alpha)$ . By the definition of semantics for  $\alpha$ , we obtain that there is  $j$  with  $i \leq j \leq k$  such that  $(\Pi, i, j) \in R(\alpha_1)$  and  $(\Pi, j, k) \in R(\alpha_2)$ . Using the induction hypothesis twice, we get a state sequence  $q_{1,0} \dots q_{1,m_1}$  with  $m_1 = \frac{j-i}{2}$  and sets  $X_{1,0}, \dots, X_{1,m_1}$  in  $M_{\alpha_1}$  and a state sequence  $q_{2,0} \dots q_{2,m_2}$  with  $m_2 = \frac{k-j}{2}$  and sets  $X_{2,0}, \dots, X_{2,m_2}$  in  $M_{\alpha_2}$ , both satisfying conditions (i) to (iv). Since  $q_{f,1}$  is  $\varepsilon$ -reachable from  $q_{1,m_1}$  by (ii) and a new  $\varepsilon$ -transition was introduced from  $q_{f,1}$  to  $q_{0,2}$ ,  $q_{2,0}$  is  $\varepsilon$ -reachable from  $q_{1,m_1}$  as well. Thus,  $q_{1,m_1}$  inherits all  $\varepsilon$ - and  $\tau$ -reachable states from  $q_{2,0}$  and we can concatenate the two state sequences, leaving out  $q_{2,0}$ . One can easily see that this is indeed a proper state sequence in  $M_\alpha$  fulfilling (i) to (iii). Since the merged state sequences consist of  $m_1 + 1$  resp.  $m_2 + 1$  states and one state was removed, the new state sequence has  $m + 1 = m_1 + 1 + m_2 + 1 - 1$  states for  $m = m_1 + m_2 = \frac{j-i}{2} + \frac{k-j}{2} = \frac{k-i}{2}$ . Condition (iv) is established with condition (iv) of the initial state sequences and by recognising that the removed state was not marked with a formula. Therefore one only has to merge the sets  $X_{1,m_1}$  and  $X_{2,0}$ .

For the reverse direction, assume that there is a state sequence with sets  $X_0, \dots, X_m$  in  $M_\alpha$  satisfying (i) to (iv). Since the final state of  $M_\alpha$  is in  $Q_2$  and the initial state of  $M_\alpha$  is in  $Q_1$ , the state sequence has to transition from  $Q_1$  to  $Q_2$  at some point. Due to the form of  $\rho$ , this can only happen once by taking the  $\varepsilon$ -transition from  $q_{f,1}$  to  $q_{0,2}$ . By inserting the state  $q_{0,2}$  into the sequence where the  $\varepsilon$ -transition was taken, we obtain two state sequences  $q_{1,0} \dots q_{1,m_1}$  and  $q_{2,0} \dots q_{2,m_2}$  in the subautomata  $M_{\alpha_1}$  and  $M_{\alpha_2}$  respectively. Corresponding sets  $X_{1,0}, \dots, X_{1,m_1}$  and  $X_{2,0}, \dots, X_{2,m_2}$  are obtained by splitting the set  $X_m$  in an appropriate way. One can easily see that these state sequences fulfill conditions (i) to (iii). Their length is  $m_1 + 1$  resp.  $m_2 + 1$  for some  $m_1, m_2$  with  $m_1 + m_2 = \frac{k-i}{2}$ . Thus, there is a  $j$  such that  $m_1 = \frac{j-i}{2}$  and  $m_2 = \frac{k-j}{2}$ . As condition (iv) for the original state sequence implies condition (iv) for the new state sequences (with indices shifted by  $j - i$  for the second path), this enables us to use the induction hypothesis twice to directly obtain the semantics definition of  $(\Pi, i, k) \in R(\alpha)$ .

$\alpha = \alpha_1^*$ : Assume  $(\Pi, i, k) \in R(\alpha)$ . Using the semantics definition, we obtain indices  $i = j_0 \leq \dots \leq j_l = k$  with  $(\Pi, j_p, j_{p+1}) \in R(\alpha_1)$  for  $0 \leq p < l$ . For  $l = 0$ ,  $i = k$  has to hold and the singleton state sequence  $q_0$  with the empty set  $X_0$  establishes our claim. If  $l > 0$ , one can use the induction hypothesis on each pair  $j_p, j_{p+1}$  to obtain a state sequence in  $M_{\alpha_1}$ . For  $l = 1$ , the state sequence and sets in  $M_{\alpha_1}$  trivially translates to a state sequence and sets in  $M_\alpha$ . For  $l > 1$ , we concatenate every two consecutive state sequences just as in the previous case  $\alpha = \alpha_1 \alpha_2$ . This leaves us with the desired state sequence in  $M_\alpha$  with the same arguments as given there.

Now assume that there is a state sequence with sets  $X_0, \dots, X_m$  in  $M_\alpha$  fulfilling conditions (i) to (iv). This state sequence can take the  $\varepsilon$ -transition between  $q_{0,f}$  and  $q_{0,1}$  arbitrarily many times. If it does not do so, it is either the state sequence  $q_0$  or a state sequence in  $M_{\alpha_1}$ . The singleton state sequence  $q_0$  with

$m = 0$  implies  $i = k$  and a single index  $j = i = k$  fulfills the semantics definition of  $(\Pi, i, k) \in R(\alpha)$ . For a state sequence in  $M_{\alpha_1}$ , the claim is directly established by the induction hypothesis for two indices  $j_1 = i, j_2 = k$ . Any other state sequence can be divided into sub-sequences by inserting states  $q_{0,1}$  and splitting sets just like in the previous case  $\alpha = \alpha_1 \alpha_2$ . Conditions (i) to (iii) are easily established for these state sequences. Condition (iv) is obtained by introducing appropriate indices  $j_1, \dots, j_l$  as substitutions for  $i$  and  $k$  for each state sequence. Using the induction hypothesis on each of the state sequences leaves us with the semantics definition of  $(\Pi, i, k) \in R(\alpha)$  with indices  $j_1, \dots, j_l$ .

$\alpha = \varphi?$ : Assume that  $(\Pi, i, k) \in R(\alpha)$ . By the definition of semantics, we obtain  $i = k$  and  $\Pi[i, \infty] \models_{\mathcal{T}} \varphi$ . This directly implies that the state sequence  $q_0$  with the set  $X_0 = \{\psi\}$  in  $M_{\alpha}$  fulfills conditions (i) to (iv).

For the other direction, assume that there is a state sequence with sets  $X_0, \dots, X_m$  in  $M_{\alpha}$  fulfilling (i) to (iv). Since there are only  $\varepsilon$ -transitions, the state sequence can only be  $q_0$ . Therefore, we know that  $k = i$  (by  $m = 0$ ) and that  $\Pi[i, \infty] \models_{\mathcal{T}} \varphi$  (by (iv) and the fact that  $q_2$  can only be reached by passing through  $q_1$ , thus  $\psi \in X_0$ ), which is exactly the definition for  $(\Pi, i, k) \in R(\alpha)$ .  $\square$

*Proof (Theorem 1).* By structural induction over the structure of  $\varphi$ . In each case, let  $\Pi$  be a path assignment with  $\nu(\Pi) = (s_0, \tau_0)(s_1, \tau_1) \dots$  where  $s_j = (s_0^j, \dots, s_n^j)$  and  $\tau_j = (\sigma_0^j, \dots, \sigma_n^j)$ .

$\varphi = \mathbf{a}_{\pi_k}$ : By construction, we see that  $\mathcal{L}(\mathcal{A}_{\varphi}) = \{(s'_0, \tau'_0)(s'_1, \tau'_1) \dots \in (S^n \times \Sigma^n)^{\omega} \mid a \in L(s'_0|_k)\}$ .

On the one hand, assume that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi})$ . Then, by the above characterisation of  $\mathcal{L}(\mathcal{A}_{\varphi})$ , we have  $a \in L(s_k^0)$  which by definition of  $\nu$  translates to  $a \in L(\Pi(\pi_k)(0))$ . Therefore, we have  $\Pi \models_{\mathcal{T}} \varphi$ .

On the other hand, assume  $\Pi \models_{\mathcal{T}} \varphi$ . By the definition of  $\varphi$ 's semantics, this implies  $a \in L(\Pi(\pi_k)(0))$  which by definition of  $\nu$  translates to  $a \in L(s_k^0)$ . Therefore, we have  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi})$  by the above characterisation of  $\mathcal{L}(\mathcal{A}_{\varphi})$ .

$\varphi = \neg \mathbf{a}_{\pi_k}$ : analogous to  $\mathbf{a}_{\pi_k}$ .

$\varphi = \varphi_1 \wedge \varphi_2$ : By construction, we have  $\mathcal{L}(\mathcal{A}_{\varphi}) = \mathcal{L}(\mathcal{A}_{\varphi_1}) \cap \mathcal{L}(\mathcal{A}_{\varphi_2})$ .

On the one hand, assume that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi})$ . By the above characterisation of  $\mathcal{L}(\mathcal{A}_{\varphi})$ , we have that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi_1})$  and  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi_2})$ . Using the induction hypothesis twice, we get that  $\mathcal{A}_{\varphi_i}$  is  $\mathcal{T}$ -equivalent to  $\varphi_i$  for  $i = 1, 2$ . Therefore,  $\Pi \models_{\mathcal{T}} \varphi_1$  and  $\Pi \models_{\mathcal{T}} \varphi_2$  which implies  $\Pi \models_{\mathcal{T}} \varphi$  by  $\varphi$ 's semantics.

On the other hand, assume that  $\Pi \models_{\mathcal{T}} \varphi$ . By the definition of  $\varphi$ 's semantics we have  $\Pi \models_{\mathcal{T}} \varphi_1$  and  $\Pi \models_{\mathcal{T}} \varphi_2$ . Using the induction hypothesis twice we get that  $\mathcal{A}_{\varphi_i}$  is  $\mathcal{T}$ -equivalent to  $\varphi_i$  for  $i = 1, 2$ . Therefore, we have that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi_1})$  and  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi_2})$  which by the above characterisation of  $\mathcal{L}(\mathcal{A}_{\varphi})$  implies  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi})$ .

$\varphi = \varphi_1 \vee \varphi_2$ : By construction, we see that  $\mathcal{L}(\mathcal{A}_{\varphi}) = \mathcal{L}(\mathcal{A}_{\varphi_1}) \cup \mathcal{L}(\mathcal{A}_{\varphi_2})$ .

On the one hand, assume that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi})$ . By the above characterisation of  $\mathcal{L}(\mathcal{A}_{\varphi})$ , we have that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi_1})$  or  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi_2})$ . We discriminate the cases and assume the former. The latter case is analogous. Then, by induction

hypothesis, since  $\mathcal{A}_{\varphi_1}$  is  $\mathcal{T}$ -equivalent to  $\varphi_1$ , we get  $\Pi \models_{\mathcal{T}} \varphi_1$  which implies  $\Pi \models_{\mathcal{T}} \varphi$  by the definition of  $\varphi$ 's semantics.

On the other hand, assume that  $\Pi \models_{\mathcal{T}} \varphi$ . By the definition of  $\varphi$ 's semantics, we get  $\Pi \models_{\mathcal{T}} \varphi_1$  or  $\Pi \models_{\mathcal{T}} \varphi_2$ . We discriminate the cases and assume the former. The other case is analogous. Then, by induction hypothesis, since  $\mathcal{A}_{\varphi_1}$  is  $\mathcal{T}$ -equivalent to  $\varphi_1$ , we get  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi_1})$  which shows the claim by the above characterisation of  $\mathcal{L}(\mathcal{A}_{\varphi})$ .

$\varphi = \langle \alpha \rangle \varphi_1$ : By construction,  $\mathcal{L}(\mathcal{A}_{\varphi})$  can be characterised as follows:  $w = (s'_0, \tau'_0)(s'_1, \tau'_1) \dots \in \mathcal{L}(\mathcal{A}_{\varphi})$  iff there is  $k$  such that:

- (i) there is a path  $q_0 q_1 \dots q_k$  from the initial state  $q_{0,\alpha}$  to a  $q_k$  with  $q_k \xrightarrow{\varepsilon}_X q_{f,\alpha}$  for some  $X$  obtained by reading  $\tau'_0 \tau'_1 \dots \tau'_{k-1}$
- (ii) for every state  $q_j$  with  $j < k$  on the path from (i),  $\psi_l \in X$  with  $q_j \xrightarrow{\tau}_X q_{j+1}$  implies  $(s'_j, \tau'_j)(s'_{j+1}, \tau'_{j+1}) \dots \in \mathcal{L}(\mathcal{A}_{\psi_l})$
- (iii)  $\psi_l \in X$  with  $q_k \xrightarrow{\varepsilon}_X q_{f,\alpha}$  implies  $(s'_k, \tau'_k)(s'_{k+1}, \tau'_{k+1}) \dots \in \mathcal{L}(\mathcal{A}_{\psi_l})$
- (iv)  $(s'_k, \tau'_k)(s'_{k+1}, \tau'_{k+1}) \dots \in \mathcal{L}(\mathcal{A}_{\varphi_1})$

On the one hand, assume that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_{\varphi})$ . By using the induction hypothesis on all subautomata of  $\mathcal{A}_{\varphi}$ , we get that they all are  $\mathcal{T}$ -equivalent to their respective formula. With the above characterisation and the  $\mathcal{T}$ -equivalences, we get that there is a  $k$  such that:

- (i) there is a path  $q_0 q_1 \dots q_k$  from  $q_{0,\alpha}$  to a  $q_k$  with  $q_k \xrightarrow{\varepsilon}_X q_{f,\alpha}$  for some  $X$  obtained by reading  $\tau_0 \dots \tau_{k-1}$  and
- (ii) for every state  $q_j$  with  $j < k$  on the path from (i),  $\psi_l \in X$  with  $q_j \xrightarrow{\tau}_X q_{j+1}$  implies  $\Pi[j, \infty] \models_{\mathcal{T}} \psi_l$
- (iii)  $\psi_l \in X$  with  $q_k \xrightarrow{\varepsilon}_X q_{f,\alpha}$  implies  $\Pi[k, \infty] \models_{\mathcal{T}} \psi_l$
- (iv)  $\Pi[k, \infty] \models_{\mathcal{T}} \varphi_1$

Since the path in (i) is a state sequence in  $M_{\alpha}$  and has appropriate length, we can use Lemma 1 to obtain that  $(\Pi, 0, k) \in R(\alpha)$ . This, together with (iv), implies  $\Pi \models_{\mathcal{T}} \varphi$ .

On the other hand, assume that  $\Pi \models_{\mathcal{T}} \varphi$ . By using the induction hypothesis on all subformulas of  $\varphi$ , we get that they all are  $\mathcal{T}$ -equivalent to their respective automaton. The definition of  $\varphi$ 's semantics implies that there is a  $k$  such that  $(\Pi, 0, k) \in R(\alpha)$  and  $\Pi[k, \infty] \models_{\mathcal{T}} \varphi_1$ . Using Lemma 1, we obtain a state sequence  $q_0 \dots q_m$  with  $m = \frac{k}{2}$  and sets  $X_0, \dots, X_m$  in  $M_{\alpha}$  such that

- (i)  $q_0$  is the initial state of  $M_{\alpha}$ ,
- (ii)  $q_m \xrightarrow{\varepsilon}_{X_m} q_{f,\alpha}$  for the final state  $q_{f,\alpha}$  of  $M_{\alpha}$ ,
- (iii)  $q_l \xrightarrow{\tau_{2l+1}}_{X_l} q_{l+1}$  for all  $l < m$ ,
- (iv)  $\psi_i \in X_l$  implies  $\Pi[2l, \infty] \models_{\mathcal{T}} \psi_i$  for all  $l \leq m$ .

Using the  $\mathcal{T}$ -equivalence of  $\varphi$ 's subformulas with their respective automata, we can translate these conditions such that there is  $k$  with:

- (i) there is a path  $q_0 q_1 \dots q_k$  from the initial state  $q_{0,\alpha}$  to a  $q_k$  with  $q_k \xrightarrow{\varepsilon}_X q_{f,\alpha}$  for some  $X$  obtained by reading  $\tau_0 \dots \tau_{k-1}$
- (ii) for every state  $q_j$  with  $j < k$  on the path,  $\psi_l \in X$  with  $q_j \xrightarrow{\tau}_X q_{j+1}$  implies  $\nu(\Pi[j, \infty]) \in \mathcal{L}(\mathcal{A}_{\psi_l})$
- (iii)  $\psi_l \in X$  with  $q_k \xrightarrow{\varepsilon}_X q_{f,\alpha}$  implies  $\nu(\Pi[k, \infty]) \in \mathcal{L}(\mathcal{A}_{\psi_l})$

(iv)  $\nu(\Pi[k, \infty]) \in \mathcal{L}(\mathcal{A}_{\varphi_1})$

By the above definition of  $\mathcal{L}(\mathcal{A}_\varphi)$ , this implies that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_\varphi)$ .

$\varphi = [\alpha]\varphi_1$ : By construction,  $\mathcal{L}(\mathcal{A}_\varphi)$  can be characterised as follows:  $w = (s'_0, \tau'_0)(s'_1, \tau'_1) \dots \in \mathcal{L}(\mathcal{A}_\varphi)$  iff

(\*) for all paths  $q_0 q_1 \dots q_k$  with (i)  $q_0 = q_{0,\alpha}$  and  $q_k \xrightarrow{\varepsilon}_{X_k} q_{f,\alpha}$  (ii)  $q_j \xrightarrow{\tau'_j}_{X_j} q_{j+1}$  for all  $j < k$  (iii)  $\psi_l \notin X_j$  or  $(s'_j, \tau'_j)(s'_{j+1}, \tau'_{j+1}) \dots \notin \mathcal{L}(\mathcal{A}_{\neg\psi_l})$  for all  $j \leq k$  we have  $(s'_k, \tau'_k)(s'_{k+1}, \tau'_{k+1}) \dots \in \mathcal{L}(\mathcal{A}_{\varphi_1})$

On the one hand, assume that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_\varphi)$ . By using the induction hypothesis on all subautomata of  $\mathcal{A}_\varphi$ , we conclude that they all are  $\mathcal{T}$ -equivalent to their respective formula. With the above characterisation of  $\mathcal{L}(\mathcal{A}_\varphi)$  and making use of  $\mathcal{T}$ -equivalences, we obtain that:

(\*) for all paths  $q_0 q_1 \dots q_k$  with (i)  $q_0 = q_{0,\alpha}$  and  $q_k \xrightarrow{\varepsilon}_{X_k} q_{f,\alpha}$  (ii)  $q_j \xrightarrow{\tau_j}_{X_j} q_{j+1}$  for all  $j < k$  (iii)  $\psi_l \in X_j$  implies  $\Pi[j, \infty] \not\models_{\mathcal{T}} \neg\psi_l$  for all  $j \leq k$  we have  $\Pi[k, \infty] \models_{\mathcal{T}} \varphi_1$

Eliminating the double negation in (iii) of (\*), we can use Lemma 1 to obtain that for all  $k$ ,  $(\Pi, 0, k) \in R(\alpha)$  implies  $\Pi[k, \infty] \models_{\mathcal{T}} \varphi_1$ , which establishes the claim.

On the other hand, assume that  $\Pi \models_{\mathcal{T}} \varphi$ . By using the induction hypothesis on all subformulas of  $\varphi$ , we get that they all are  $\mathcal{T}$ -equivalent to their respective automaton. The definition of  $\varphi$ 's semantics implies that for all  $k$ ,  $(\Pi, 0, k) \in R(\alpha)$  implies  $\Pi[k, \infty] \models_{\mathcal{T}} \varphi$ . Using Lemma 1, we can reformulate the premise of the implication to be the existence of a state sequence  $q_0 \dots q_k$  in  $M_\alpha$  such that  $q_0$  is the initial state,  $q_j \xrightarrow{\tau_j}_{X_j} q_{j+1}$  for  $j < k$ ,  $q_k \xrightarrow{\varepsilon}_{X_k} q_{f,\alpha}$  for the final state  $q_{f,\alpha}$  of  $M_\alpha$  and for all  $j \leq k$ ,  $\psi_l \in X_j$  implies  $\Pi[j, \infty] \models_{\mathcal{T}} \psi_l$ . Making use of  $\mathcal{T}$ -equivalences and introducing a double negation, we get that:

(\*) for all paths  $q_0 q_1 \dots q_k$  with (i)  $q_0 = q_{0,\alpha}$  and  $q_k \xrightarrow{\varepsilon}_{X_k} q_{f,\alpha}$  (ii)  $q_j \xrightarrow{\tau'_j}_{X_j} q_{j+1}$  for all  $j < k$  (iii)  $\psi_l \notin X_j$  or  $\nu(\Pi[j, \infty]) \dots \notin \mathcal{L}(\mathcal{A}_{\neg\psi_l})$  for all  $j \leq k$  we have  $\nu(\Pi[k, \infty]) \dots \in \mathcal{L}(\mathcal{A}_{\varphi_1})$

This, by the above characterisation of  $\mathcal{L}(\mathcal{A}_\varphi)$  implies  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_\varphi)$ .

$\varphi = \Delta\alpha$ : By construction,  $\mathcal{L}(\mathcal{A}_\varphi)$  can be characterised as follows:  $w = (s'_0, \tau'_0)(s'_1, \tau'_1) \dots \in \mathcal{L}(\mathcal{A}_\varphi)$  iff

(i) there are infinitely many indices  $0 = k_1 \leq k_2 \leq \dots$  such that for all  $i$ ,  $q_{k_i} = q_0$  and there is a path  $q_{k_i} \dots q_{k_{i+1}}$  induced by a state sequence  $q_0 q_1 \dots q_m$

with  $q_0 = q_{0,\alpha}$ ,  $q_j \xrightarrow{\tau'_{k_i+j}}_{X_j} q_{j+1}$  for  $j < m$  and  $q_m \xrightarrow{\varepsilon}_{X_m} q_{f,\alpha}$  and

(ii) for each  $0 \leq j \leq m$  in such a state sequence  $\psi_l \in X_j$  implies

$(s'_{k_i+j}, \tau'_{k_i+j})(s'_{k_i+j+1}, \tau'_{k_i+j+1}) \dots \in \mathcal{L}(\mathcal{A}_{\psi_l})$

On the one hand, assume that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_\varphi)$ . By using the induction hypothesis on all subautomata of  $\mathcal{A}_\varphi$ , we see that they all are  $\mathcal{T}$ -equivalent to their respective formula. The above characterisation of  $\mathcal{L}(\mathcal{A}_\varphi)$  implies that:

(i) there are infinitely many indices  $0 = k_1 \leq k_2 \leq \dots$  such that for all  $i$ ,  $q_{k_i} = q_0$  and there is a path  $q_{k_i} \dots q_{k_{i+1}}$  induced by a state sequence  $q_0 q_1 \dots q_m$

with  $q_0 = q_{0,\alpha}$ ,  $q_j \xrightarrow{\tau'_{k_i+j}}_{X_j} q_{j+1}$  for  $j < m$  and  $q_m \xrightarrow{\varepsilon}_{X_m} q_{f,\alpha}$  and

(ii) for each  $0 \leq j \leq m$  in such a state sequence  $\psi_l \in X_j$  implies  $\nu(\Pi[k_i +$

$j, \infty]) \dots \in \mathcal{L}(\mathcal{A}_{\psi_l})$

Using  $\mathcal{T}$ -equivalences, we can translate (ii) to a form where Lemma 1 is applicable to the paths in (i) and (ii) to directly obtain the semantics definition of  $\Pi \models_{\mathcal{T}} \varphi$ .

On the other hand, assume that  $\Pi \models_{\mathcal{T}} \varphi$ . Using the induction hypothesis on all subformulas of  $\varphi$ , we get that they all are  $\mathcal{T}$ -equivalent to their respective automaton. The definition of  $\varphi$ 's semantics implies that there are infinitely many indices  $0 = k_1 \leq k_2 \leq \dots$  such that for all  $i \geq 1$ ,  $(\Pi, k_i, k_{i+1}) \in R(\alpha)$  holds. For each of these triples, we can use Lemma 1 to receive a state sequence  $q_0 \dots q_m$  with sets  $X_0, \dots, X_m$  in  $M_\alpha$  such that  $q_0 = q_{o,\alpha}$ ,  $q_j \xrightarrow{\tau_{k_i+j}}_{X_j} q_{j+1}$  for  $j < m$  and  $q_k \xrightarrow{\epsilon}_{X_m} q_{f,\alpha}$  for the final state  $q_{f,\alpha}$  of  $M_\alpha$  where  $\psi_l \in X_j$  implies  $\Pi[k_i+j, \infty] \models_{\mathcal{T}} \psi_l$ . By replacing the first and last state of each sequence with  $q_0$  and using  $\mathcal{T}$ -equivalences, we directly obtain the above characterisation for  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_\varphi)$ .

$\varphi = \neg \Delta \alpha$ : This case is directly implied by the induction hypothesis and Proposition 2.

$\varphi = \exists \pi. \varphi_1$ : Since  $\mathcal{A}_\varphi$  is a Büchi automaton, its language can be characterised such that  $w \in \mathcal{L}(\mathcal{A}_\varphi)$  iff there is a path  $q_0(q_1, s_1, \sigma_1)(q_2, s_2, \sigma_2) \dots$  in  $\mathcal{A}_\varphi$  witnessing the acceptance of  $w$ .

On the one hand, assume that  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_\varphi)$ . By using the induction hypothesis on  $\mathcal{A}_{\varphi_1}$ , we get that it is  $\mathcal{T}$ -equivalent to  $\varphi_1$ . Let  $q_0(q_1, s_1, \sigma_1)(q_2, s_2, \sigma_2) \dots$  be the path in  $\mathcal{A}_\varphi$  witnessing the acceptance of  $\nu(\Pi)$  as in the above characterisation. Note that by construction of  $\mathcal{A}_\varphi$ ,  $p = s_0 \sigma_0 s_1 \sigma_1 \dots$  for appropriate  $s_0, \sigma_0$  is also a path of  $\mathcal{T}$ , starting at  $\Pi(\epsilon) = s|_n$ , or more formally,  $p \in \text{Paths}(\mathcal{T}, \Pi(\epsilon)(0))$ . Let  $\Pi'$  be the path assignment  $\Pi[\pi_{n+1} \rightarrow p, \epsilon \rightarrow p]$ . Then, the construction also makes sure that  $\nu(\Pi') \in \mathcal{L}(\mathcal{A}_{\varphi_1})$ , which implies that  $\Pi' \models_{\mathcal{T}} \varphi_1$  by the use of a  $\mathcal{T}$ -equivalence. Therefore, we directly obtain  $\Pi \models_{\mathcal{T}} \varphi$ .

On the other hand, assume that  $\Pi \models_{\mathcal{T}} \varphi$ . By using the induction hypothesis on  $\varphi_1$ , we get that it is  $\mathcal{T}$ -equivalent to  $\mathcal{A}_{\varphi_1}$ . The definition of  $\varphi$ 's semantics implies that there is a path  $p \in \text{Paths}(\mathcal{T}, \Pi(\epsilon)(0))$  such that  $\Pi' := \Pi[\pi \rightarrow p, \epsilon \rightarrow p] \models_{\mathcal{T}} \varphi_1$ . Using  $\mathcal{T}$ -equivalence, we obtain that  $\nu(\Pi') \in \mathcal{L}(\mathcal{A}_{\varphi_1})$ . The fact that we use a dealternised version of  $\mathcal{A}_{\varphi_1}$  equips us with a path  $q_0 q_1 \dots$  in  $\mathcal{A}_{\varphi_1}$  witnessing the acceptance of  $\nu(\Pi')$ . Since  $p = s_0 \sigma_0 s_1 \sigma_1 \dots$  is a path in  $\mathcal{T}$ , starting at  $\Pi(\epsilon) = s|_n$ , we can construct a path  $q_0(q_1, s_1, \sigma_1)(q_2, s_2, \sigma_2) \dots$  in  $\mathcal{A}_\varphi$  witnessing the acceptance of  $\nu(\Pi)$  in  $\mathcal{A}_\varphi$  which by the above characterisation implies  $\nu(\Pi) \in \mathcal{L}(\mathcal{A}_\varphi)$ .

$\varphi = \neg \exists \pi. \varphi_1$ : This case is directly implied by the induction hypothesis, Proposition 1 and Proposition 2.  $\square$

*Proof (Lemma 2).* By induction on the criticality  $k$ .

**Base case:**  $\varphi$  has criticality 0. We show that  $\mathcal{A}_\varphi$  has size  $2^{\mathcal{O}(p(n))} \cdot p'(m)$  in the size  $n$  of  $\varphi$  and the size  $m$  of  $\mathcal{T}$  for some polynomials  $p, p'$ . Let us inductively show this claim.

First, notice that  $|M_\alpha|$  is linear in the size of  $\alpha$ . This can easily be shown by a structural induction, where each construction adds a constant number of states to its subautomata only.



Basic constructions  $\mathcal{A}_{a_\pi}$  and  $\mathcal{A}_{\neg a_\pi}$  have constant size. For boolean connectives as well as  $\langle \alpha \rangle \varphi$ ,  $[\alpha] \varphi$  and  $\Delta \alpha$ , the construction of  $\mathcal{A}_\varphi$  again just adds a constant number of states to the automata for the subformulas. The construction for  $\neg \Delta \alpha$  introduces a quadratic increase by Proposition 2. Over the course of the whole construction, this results in an exponential increase in the nesting depth of negated  $\Delta \alpha$  constructs at most. To be more precise, when bounding this nesting depth to a constant  $d$ , the polynomial  $p$  inducing that our construction has size  $g_{p,c}(k, n)$  has degree at most  $2d$ . Existential quantifiers increase the size of the automaton exponentially and add a factor polynomial in the size of the structure  $\mathcal{T}$ . Note that the factor depending on  $|\mathcal{T}|$  is added after the exponential blowup due to the dealternation construction  $MH(\mathcal{A})$ .

It remains to show that the dealternation construction  $MH(\mathcal{A})$  introduces an exponential blowup of the structure's size at most once for formulas of criticality 0, regardless of how many quantifiers the formula contains. In order to do this, we have to look closer at the proof of Proposition 1. We show that once the dealternation construction  $MH(\mathcal{A})$  is done for the innermost quantifiers, at most one state of each dealternised automaton has to be tracked at the same time in further dealternations. Thus, the exponential size of the subautomaton is added as a factor rather than in an exponent when determining the size of the state space of the full automaton.

Our claim can be shown by an induction over the number of constructions on top of the dealternised automaton. In the base case, no construction is done on top of a dealternised automaton  $\mathcal{A}_\varphi$ . Since  $\mathcal{A}_\varphi$  is a Büchi automaton, a run of the resulting automaton is a path rather than a tree on every word. Thus, only a single state has to be tracked at the same time.

In the inductive step, we discriminate cases for the outermost construction. By the induction hypothesis, at most one state of each dealternised automaton has to be tracked in each subautomaton. For the construction  $\varphi_1 \vee \varphi_2$ , a run tree moving into  $\mathcal{A}_{\varphi_1}$  or  $\mathcal{A}_{\varphi_2}$  never returns to the initial state. Thus, since  $\mathcal{A}_{\varphi_1}$  and  $\mathcal{A}_{\varphi_2}$  are unconnected, we track states of only one of the automata. Then, the claim is implied by the induction hypothesis. The construction for  $\mathcal{A}_{\varphi_1 \wedge \varphi_2}$  works similarly, with the difference that we have to track states of both subautomata when a run moves into this automaton over the initial state. This does, however, not lead to an increase in states of each dealternised automaton that have to be tracked, since these subautomata are unconnected. The next construction we have to consider is  $\mathcal{A}_{\langle \alpha \rangle \varphi}$ . Here, a run has the property that at most one state of  $M_\alpha$  has to be tracked, which can be replaced by states of  $\mathcal{A}_\varphi$  at some point. Additionally, arbitrarily many states of  $\mathcal{A}_\psi$  for subformulas  $\psi$  of  $\alpha$  can be tracked. However, this is no contradiction to our claim, since  $\alpha$  may not contain any quantified subformulas and thus  $\mathcal{A}_\psi$  may not contain dealternised automata in uncritical formulas. Thus, since the induction hypothesis states that at most one state of each dealternised automaton of  $\mathcal{A}_\varphi$  has to be tracked at any point, this shows our claim. As the last case, we consider the construction for  $\exists \pi. \varphi$ . Here, since only a disjunctive transition is added on top of  $\mathcal{A}_\varphi$ , we can argue similar as in the case for  $\varphi_1 \vee \varphi_2$  with the difference that we consider only a

single subautomaton. Due to the exemption rule in the definition of criticality, there is an additional construction that has to be considered:  $[\alpha]\varphi$ , where  $\alpha$  is deterministic<sup>2</sup> and the outermost quantifier inside  $\alpha$  is negated. Since tests  $\psi$  in  $\alpha$  are handled by disjunctively transitioning into the automaton for  $\neg\psi$  in the  $[\alpha]\varphi$  construction, this cancels out the negation of the quantifier. Therefore, no critical negation construction has to be performed on a dealternised subautomaton during the construction of  $\mathcal{A}_{\neg\psi}$ . Then, since due to the fact that  $M_\alpha$  is deterministic, conjunctions of transitions in  $M_\alpha$  behave the same as disjunctions and we can argue just as in the case for  $\mathcal{A}_{\langle\alpha\rangle\varphi}$ . Finally, observe that we do not have to consider constructions for  $\Delta\alpha$ ,  $\neg\Delta\alpha$ , or general  $[\alpha]\varphi$ , since the resulting formula is not uncritical when any of these contain a quantified subformula.

**Inductive step:**  $\varphi$  has criticality  $k+1$ . On the path in  $\varphi$ 's syntax tree inducing the criticality, we inspect the outermost critical quantifier. Its subformulas  $\varphi_i$  have criticality at most  $k$ . Using the induction hypothesis on all subformulas, we obtain automata of size at most  $\mathcal{O}(g(k+1, |\varphi_i|) \cdot g(k, |\mathcal{T}|))$ . Going on with the construction, the next dealternation will result in an automaton of size  $2^{\mathcal{O}(|\mathcal{A}_{\varphi_i}|)}$  (by Proposition 1) which can be bounded by  $2^{(\mathcal{O}(g(k+1, |\varphi_i|)) \cdot g(k, |\mathcal{T}|))} = \mathcal{O}(g(k+2, |\varphi|) \cdot g(k+1, |\mathcal{T}|))$ . Since we inspected the outermost critical quantifier on the path inducing the criticality of the formula, any of the subsequent constructions will not cause a further exponential blowup of the automaton's size, as argued in the base case.  $\square$

*Proof (Theorem 3).* We reduce the model checking problem of alternation depth  $k$  HyperCTL\*-formulas, which by [3] is hard for  $\text{NSPACE}(g(k, |\varphi|))$  and for  $\text{NSPACE}(g(k-1, |\mathcal{T}|))$ , to the criticality  $k$  model checking problem in our logic.

Given a Kripke structure  $K$  and a HyperCTL\* formula  $\varphi$  with alternation depth  $k$ , we translate them to a  $KTS$   $tr(K)$  and a HyperPDL- $\Delta$  formula  $tr(\varphi)$  of criticality  $k$  such that  $K \models \varphi$  iff  $tr(K) \models tr(\varphi)$ .

The translation of the Kripke structure  $K = (S, s_0, \delta, L)$  is the structure itself, seen as a  $KTS$ : We keep  $S, s_0, L$  and define  $\delta_\sigma := \delta$  for  $\Sigma = \{\sigma\}$ . If one wants to define the  $KTS$  over a larger set  $\Sigma$ , this would be possible by choosing the same  $\delta_\sigma$  for all  $\sigma \in \Sigma$ . This would however unnecessarily complicate this proof, since the set of transitions that can be taken is not affected by the choice of atomic programs  $\sigma$  in any state  $s \in S$ .

For formulas  $\varphi$ , only the cases  $\bigcirc\varphi$ ,  $\varphi_1\mathcal{U}\varphi_2$  and  $\varphi_1\mathcal{R}\varphi_2$  are non-trivial. We translate them in the following way:  $tr(\bigcirc\varphi) := \langle\bullet\rangle tr(\varphi)$  or  $tr(\bigcirc\varphi) := [\bullet]tr(\varphi)$ , such that  $\langle\bullet\rangle tr(\varphi)$  is used in the negation normal form (this way a translation of  $\bigcirc\varphi$  never induces criticality on its own). Furthermore, we translate  $tr(\varphi_1\mathcal{U}\varphi_2) := \langle(tr(\varphi_1)?\bullet)^*\rangle tr(\varphi_2)$  and  $tr(\varphi_1\mathcal{R}\varphi_2) := [(tr(\neg\varphi_1)?\bullet)^*]tr(\varphi_2)$ . Since

<sup>2</sup> There are additional forms of  $\alpha$ , where a negated quantifier inside the modality  $[\alpha]$  is uncritical. This includes all forms, where in any run in  $M_\alpha$ , a test for  $\neg\psi$  occurs only such that all states occurring at the same level of the run can transition into  $\mathcal{A}_{\neg\psi}$ . Then, when a disjunctive transition into  $\mathcal{A}_{\neg\psi}$  can be taken in one of the states, it can be taken in all of the states. Since they are on the same level, the same continuation can be used for all these branches, thus only having to track a single state of each dealternised subautomaton.

existential and universal quantifiers are not changed,  $tr(\bigcirc\varphi)$  does not contain a  $\psi?$  construct, and  $\mathcal{U}$  (resp.  $\mathcal{R}$ ) modalities are translated to a single use of a  $\langle\alpha\rangle\varphi$  (resp.  $[\alpha]$ ) construct with use of  $\psi?$  where  $\varphi_i$  and  $tr(\varphi_i)$  have the same role in the definition of alternation depth and criticality,  $\varphi$  and  $tr(\varphi)$  correspond in alternation depth resp. criticality. Note, that in the case of  $\mathcal{R}$  this only holds due to the exemption of negated outermost quantifiers in tests occurring in modalities  $[\alpha]$  for deterministic  $\alpha$ .

We now show that  $K \models \varphi$  iff  $tr(K) \models tr(\varphi)$  indeed holds. First, notice that a path assignment  $\Pi$  on  $K$  with  $\Pi(\pi_i) = s_0^i s_1^i \dots$  can be directly translated to a path assignment  $tr(\Pi)$  on  $tr(K)$  with  $tr(\Pi)(\pi_i) = s_0^i \sigma s_1^i \sigma \dots$ . Then, the assumption immediately follows from the statement  $\Pi[i, \infty] \models_K \varphi$  iff  $tr(\Pi)[2i, \infty] \models_{tr(K)} tr(\varphi)$  which can be shown by structural induction on  $\varphi$ . In all cases where the translation  $tr$  is trivial, the proof is as well. Let us consider the non-trivial cases:

$\bigcirc\varphi$ : Note that for  $\alpha = \bullet = (\cdot, \dots, \cdot)$ ,  $(\Pi, 0, 2) \in R(\alpha)$  trivially holds for any path assignment  $\Pi$ . Thus, checking  $tr(\Pi) \models_{tr(K)} tr(\bigcirc\varphi)$  reduces to checking whether  $tr(\Pi)[2, \infty] \models_{tr(K)} tr(\varphi)$  holds. Then, the assumption immediately follows from the induction hypothesis.

$\varphi_1 \mathcal{U} \varphi_2$ : Using our semantics definition and our above argument about wildcard programs in  $\alpha$ , the model checking problem for  $\langle(tr(\varphi_1)?\bullet)^*\rangle tr(\varphi_2)$  translates to

$$\begin{aligned} \exists i \geq 0. tr(\Pi)[i, \infty] \models_{tr(K)} tr(\varphi_2) \wedge \\ \exists n \geq 0, 0 = k_1 \leq k_2 \leq \dots \leq k_n = i. \\ \forall 1 \leq l < n. k_{i+1} = k_i + 2 \wedge tr(\Pi)[k_l, \infty] \models_{tr(K)} tr(\varphi_1) \end{aligned}$$

In this condition, notice that both  $i$  and all  $k_l$  are even. Together with the structure of  $tr(\Pi)$ , this enables us to use the induction hypothesis on  $tr(\Pi)[i, \infty] \models_{tr(K)} tr(\varphi_2)$  and  $tr(\Pi)[k_l, \infty] \models_{tr(K)} tr(\varphi_1)$  to show that the above translation is equivalent to the semantics of  $\mathcal{U}$  on  $\Pi$ , which shows the assumption.

$\varphi_1 \mathcal{R} \varphi_2$ : similar to case  $\varphi_1 \mathcal{U} \varphi_2$  □

## B Missing proofs from Section 6

*Proof (Theorem 9, encoding into MSO[E]).* We construct an inductive translation, using the induction hypothesis that (i) for every program  $\alpha$ , there is an MSO[E] formula  $\psi_\alpha(\Pi, v_i, v_j)$  equivalent to  $(\Pi, i, j) \in R(\alpha)$  when  $i$  and  $j$  are the indices of  $v_i$  and  $v_j$  on  $\pi_1$  in  $\Pi$  and (ii) for every HyperPDL- $\Delta$  formula  $\varphi$ , there is an equivalent MSO[E] formula  $tr(\varphi)$ . Atomic propositions, logical conjunctions and disjunctions and negations are trivial and quantified paths can be translated exactly as in the translation of HyperQCTL\* to MSO[E] in [5]. We therefore only consider regular expressions. These can be translated analogous to Büchi's classical proof of the equivalence of MSO and regular expressions on finite words with the only difference being that multiple paths (as sets) are considered simultaneously and synchronised with the  $E$  predicate. We describe

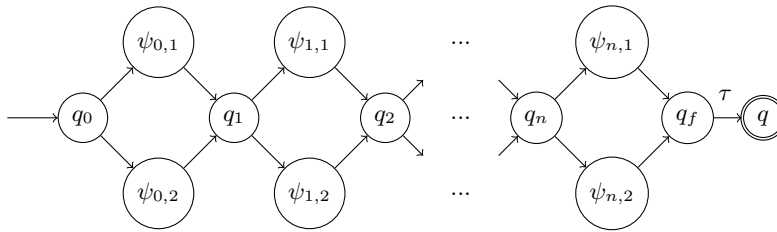
the case of  $\Delta\alpha$  only and for all other cases, similar modifications of the Büchi approach can be used.  $\Delta\alpha$  is translated as

$$\begin{aligned} \exists X_{\pi_1} \dots \exists X_{\pi_n} \cdot & \bigwedge_{1 \leq i \leq n} (X_{\pi_i} \subseteq \pi_i \wedge E(\min(X_{\pi_i}), \min(X_{\pi_1})) \\ & \wedge \forall k \in X_{\pi_1} \cdot \forall l \in X_{\pi_i} E(k, l) \Rightarrow E(\text{next}(X_{\pi_1}, k), \text{next}(X_{\pi_i}, l))) \\ & \wedge \forall v_1 \in X_{\pi_1} \dots \forall v_n \in X_{\pi_n} \cdot \\ & (\bigwedge_{1 \leq i \leq n} E(v_1, v_i)) \Rightarrow \psi_\alpha(\Pi, v_1, \text{next}(X_{\pi_1}, v_1)). \end{aligned}$$

Here,  $\min(X)$  (denoting the minimal element of  $X$ ) and  $\text{next}(X, v)$  (denoting the next element in  $X$  after  $v$ ) are used in a functional notation, but can obviously be converted to a purely predicative notation. Intuitively, the sets  $X_{\pi_1}, \dots, X_{\pi_n}$  are the subsets of nodes from  $\pi_1, \dots, \pi_n$ , where indices  $k_1, k_2, \dots$  from the definition of semantics are matched.  $\square$

## C Alternative Construction for the transition function of $\mathcal{A}_\varphi$

In our paper, we have argued that the transition function in  $\mathcal{A}_\varphi$  for  $\varphi$  containing  $\alpha$  can be exponential in the size of  $\alpha$ . Consider the automaton in Figure 5 where states annotated with  $\psi_{i,j}$  are marked with the corresponding formula and each unannotated edge stands for an  $\varepsilon$ -transition. Such an automaton can occur when  $\alpha$  has the form  $(\psi_{0,1}? + \psi_{0,2}?) (\psi_{1,1}? + \psi_{1,2}?) \dots (\psi_{n,1}? + \psi_{n,2}?) \tau$  and should serve as an illustrative example. We consider the case where this  $\alpha$  is used in a  $\langle . \rangle$  formula. For ease of presentation, we assume that  $\psi_{i,j}$  can be tested by moving into a state  $p_{i,j}$ . It is possible to reach  $q_f$  from  $q_0$  with an exponential number of  $\varepsilon$ -paths, each with a different combination of markings. Therefore we have  $q_0 \xrightarrow{\tau}_X q$  for exponentially many  $X$ , transferring into the size of the transition function when constructing  $\rho(q_0, \tau) \equiv \bigvee \{q \wedge \bigwedge_{\psi_{i,j} \in X} p_{i,j} \mid q_0 \xrightarrow{\tau}_X q\}$ .



**Fig. 5.** Automaton  $M_\alpha$  with an exponential number of test combinations

For this example, it is easy to see that these exponentially many different combinations of transitions could equally be represented by a formula of a much smaller size, namely  $(p_{0,1} \vee p_{0,2}) \wedge (p_{1,1} \vee p_{1,2}) \wedge \dots \wedge (p_{n,1} \vee p_{n,2})$ . This is due

to the fact that conjunction and disjunction closely resemble the behaviour of  $+$  and concatenation constructions in  $M_\alpha$  when considering  $\varepsilon$ -paths. We will show here, that using these ideas it is possible to construct such a formula of size not greater than  $2 \cdot |\alpha|$  for every  $\alpha$ .

We proceed by inductively constructing a function  $\varepsilon p$  such that  $\varepsilon p(q, q', \alpha) = \vartheta$  for a  $\vartheta \equiv \bigvee \{ \bigwedge_{\psi_i \in X} v_i \mid q \xrightarrow{\varepsilon}_X q' \}$  for  $\xrightarrow{\varepsilon}_X$  constructed from  $M_\alpha$ . Here we use variables  $v_i$  as placeholders for formulas  $\psi_i$  to be later replaced by a transition into the corresponding automaton  $\mathcal{A}_{\psi_i}$ . For  $\tau$ -transitions, this can straightforwardly be extended to  $\tau p$  which can then be used to construct the  $\tau$  transition function for a state  $q$  in a more succinct way. For a  $\langle \alpha \rangle \varphi$  formula and some  $q \in Q_\alpha$  we then have

$$\rho(q, (s, \tau)) = \bigvee \{ q' \wedge \tau p(q, q', \alpha) [\rho_{\psi_i}(q_0, \psi_i, (s, \tau)) / v_i] \mid q' \in Q \}.$$

Some remarks are in order for this transition function construction: (i) in this construction, opposed to the one used before, each  $q' \in Q$  can occur at most once in the disjunction, (ii) for  $q' \in Q$  such that there is no  $X$  with  $q \xrightarrow{\tau}_X q'$ , i.e.  $q'$  is not reachable from  $q$  with  $\tau$ , we have  $\tau p(q, q', \alpha) \equiv \text{false}$  and thus the state can be eliminated from the disjunction and (iii) for  $[\cdot]$  formulas,  $\tau p$  and  $\rho$  have to be constructed in the dual way by swapping  $\wedge$  and  $\vee$  operators.

Since there is no direct way to create the desired formula for  $\alpha = \alpha_1^*$  while meeting the size constraints, we construct another function  $\varepsilon p^*$  in parallel. Intuitively,  $\varepsilon p^*(q, q', \alpha)$  should have the exact properties as  $\varepsilon p(q, q', \alpha^*)$  and allows us to make different inductive constructions based on the form of  $\alpha$ . It constructs  $\varepsilon$ -paths from  $q$  to  $q'$  through  $M_\alpha$  with added  $\varepsilon$ -transitions between starting and accepting state. This allows us to amortize a formula's size increase when going over from  $\alpha$  to  $\alpha^*$  against the size of  $\alpha$ .

**Inductive construction of  $\varepsilon p$  and  $\varepsilon p^*$ :**

$$\begin{aligned} \alpha = \tau \quad \varepsilon p(q, q', \alpha) &= \begin{cases} \text{false} & \text{if } q \neq q' \\ \text{true} & \text{else} \end{cases} \\ \varepsilon p^*(q, q', \alpha) &= \text{true} \quad \text{for } q, q' \in Q \\ \alpha = \varepsilon \quad \varepsilon p(q, q', \alpha) &= \begin{cases} \text{false} & \text{if } q = q_1, q' = q_0 \\ \text{true} & \text{else} \end{cases} \\ \varepsilon p^*(q, q', \alpha) &= \text{true} \quad \text{for } q, q' \in Q \\ \alpha = \alpha_1 + \alpha_2 \quad \varepsilon p(q, q', \alpha) &= \begin{cases} \varepsilon p(q, q', \alpha_1) & \text{if } q, q' \in Q_i \\ \varepsilon p(q_0, i, q', \alpha_i) & \text{if } q = q_0, q' \in Q_i \\ \varepsilon p(q, q_{f,i}, \alpha_i) & \text{if } q \in Q_i, q' = q_f \\ \varepsilon p(q_0, 1, q_{f,1}, \alpha_1) \vee \varepsilon p(q_0, 2, q_{f,2}, \alpha_2) & \text{if } q = q_0, q' = q_f \\ \text{false} & \text{else} \end{cases} \end{aligned}$$

$$\begin{aligned}
\varepsilon p^*(q, q', \alpha) &= \begin{cases} true & \text{if } q, q' \in \{q_0, q_f\} \\ \varepsilon p(q_0, q', \alpha) & \text{if } q \in \{q_0, q_f\}, q' \in Q_1 \cup Q_2 \\ \varepsilon p(q, q_f, \alpha) & \text{if } q \in Q_1 \cup Q_2, q' \in \{q_0, q_f\} \\ \varepsilon p^*(q, q', \alpha_i) & \text{if } q, q' \in Q_i \\ \varepsilon p(q, q_{f,i}, \alpha_i) \wedge \varepsilon p(q_{0,j}, q', \alpha_j) & \text{if } q \in Q_i, q' \in Q_j, i \neq j \end{cases} \\
\alpha = \alpha_1 \alpha_2 \quad \varepsilon p(q, q', \alpha) &= \begin{cases} \varepsilon p(q, q', \alpha_i) & \text{if } q, q' \in Q_i \\ \varepsilon p(q, q_{f,1}, \alpha_1) \wedge \varepsilon p(q_{0,2}, q', \alpha_2) & \text{if } q \in Q_1 \text{ and } q' \in Q_2 \\ false & \text{else} \end{cases} \\
\varepsilon p^*(q, q', \alpha) &= \begin{cases} true & \text{if } q, q' \in \{q_{0,1}, q_{f,2}\} \\ \varepsilon p(q, q', \alpha_1) \vee (\varepsilon p^*(q, q', \alpha_1) \wedge \varepsilon p(q_{0,2}, q_{f,2}, \alpha_2)) & \text{if } q, q' \in Q_1 \\ \varepsilon p(q, q', \alpha_2) \vee (\varepsilon p(q_{0,1}, q_{f,1}, \alpha_1) \wedge \varepsilon p^*(q, q', \alpha_2)) & \text{if } q, q' \in Q_2 \\ \varepsilon p(q, q', \alpha) & \text{if } q \in Q_1, q' \in Q_2 \\ \varepsilon p(q, q_{f,2}, \alpha_2) \wedge \varepsilon p(q_{0,1}, q', \alpha_1) & \text{if } q \in Q_2, q' \in Q_1 \end{cases} \\
\alpha = (\alpha_1)^* \quad \varepsilon p(q, q', \alpha) &= \varepsilon p^*(q, q', \alpha_1) \\
\varepsilon p^*(q, q', \alpha) &= \varepsilon p^*(q, q', \alpha_1) \\
\alpha = \psi_k? \quad \varepsilon p(q, q', \alpha) &= \begin{cases} true & \text{if } q = q' \neq q_1 \\ false & \text{if } q = q_i, q' = q_j, i > j \\ v_k & \text{else} \end{cases} \\
\varepsilon p^*(q, q', \alpha) &= \begin{cases} v_k & \text{if } q = q_1 \text{ or } q' = q_1 \\ true & \text{else} \end{cases}
\end{aligned}$$

### Theoretical justification:

In order to use this succinct alternative in our construction, we have to argue that it indeed has the desired properties. Therefore we establish two theorems:

**Theorem 10.**  $Max\{|\varepsilon p(q, q', \alpha)| \mid q, q' \in Q_\alpha\} \cup \{|\varepsilon p^*(q, q', \alpha)| \mid q, q' \in Q_\alpha\} \leq 2 \cdot |\alpha|$  for all  $q, q'$  and  $\alpha$

*Proof.* Straightforward structural induction on  $\alpha$ . It is easy to see that in each case of the construction, at most two operators are added to  $\varepsilon p(q, q', \alpha)$  and  $\varepsilon p^*(q, q', \alpha)$ .

**Theorem 11.** We have  $\varepsilon p(q, q', \alpha) \equiv \bigvee \{\bigwedge_{\psi_i \in X} v_i \mid q \xrightarrow{\varepsilon}_X q'\}$  for  $\xrightarrow{\varepsilon}_X$  constructed from  $M_\alpha$  and  $\varepsilon p^*(q, q', \alpha) \equiv \bigvee \{\bigwedge_{\psi_i \in X} v_i \mid q \xrightarrow{\varepsilon}_X q'\}$  for  $\xrightarrow{\varepsilon}_X$  constructed from  $M_{\alpha^*}$ .

*Proof.* We show both claims simultaneously by a structural induction on  $\alpha$ .

**Case  $\alpha = \tau$ :** There are two unmarked states  $q_0, q_1$  in  $M_\alpha$  with a  $\tau$ -transition connecting them. There are no  $\varepsilon$ -transitions. Thus, we have  $q \xrightarrow{\varepsilon}_X q'$  iff  $q = q'$  and  $X = \emptyset$ . Therefore we have  $\bigvee \{\bigwedge_{\psi_i \in X} v_i \mid q \xrightarrow{\varepsilon}_X q'\} \equiv false$  for  $q \neq q'$  and  $\bigvee \{\bigwedge_{\psi_i \in X} v_i \mid q \xrightarrow{\varepsilon}_X q'\} \equiv true$  for  $q = q'$ , establishing the first claim.

For the second claim, consider  $\xrightarrow{\varepsilon}_X$  constructed from  $M_{\alpha^*}$ , which is obtained from  $M_{\alpha}$  by introducing  $\varepsilon$ -transition in between the starting and final state. This makes both states  $\varepsilon$ -reachable from each other, still without seeing state markings as these have not changed. With the same argument as above, we then have  $\bigvee\{\bigwedge_{\psi_i \in X} v_i | q \xrightarrow{\varepsilon}_X q'\} \equiv \text{true}$  for arbitrary  $q, q'$ , establishing the claim.

**Case  $\alpha = \varepsilon$ :** There are two unmarked states  $q_0, q_1$  in  $M_{\alpha}$  with an  $\varepsilon$ -transition connecting  $q_0$  to  $q_1$ . Thus, we have  $q \xrightarrow{\varepsilon}_X q'$  iff  $q \neq q_1, q' \neq q_0$  and  $X = \emptyset$ . Therefore we have  $\bigvee\{\bigwedge_{\psi_i \in X} v_i | q \xrightarrow{\varepsilon}_X q'\} \equiv \text{false}$  for  $q = q_1, q' = q_0$  and  $\bigvee\{\bigwedge_{\psi_i \in X} v_i | q \xrightarrow{\varepsilon}_X q'\} \equiv \text{true}$  else, establishing the first claim.

For the second claim, consider  $\xrightarrow{\varepsilon}_X$  constructed from  $M_{\alpha^*}$ , which is obtained from  $M_{\alpha}$  by introducing the missing  $\varepsilon$ -transition from  $q_1$  to  $q_0$ . This makes both states  $\varepsilon$ -reachable from each other, still without seeing state markings as these have not changed. With the same argument as above, we then have  $\bigvee\{\bigwedge_{\psi_i \in X} v_i | q \xrightarrow{\varepsilon}_X q'\} \equiv \text{true}$  for arbitrary  $q, q'$ , establishing the claim.

**Case  $\alpha = \alpha_1 + \alpha_2$ :** By induction hypothesis, the claim holds for  $\alpha_1$  and  $\alpha_2$ . To obtain  $M_{\alpha}$  from  $M_{\alpha_1}$  and  $M_{\alpha_2}$ , a new starting and final state are added with  $\varepsilon$ -transitions to the old starting states and from the old final states, respectively. We consider different cases how a path inducing  $q \xrightarrow{\varepsilon}_X q'$  could have been constructed. In the first case, where both  $q$  and  $q'$  are in the same automaton  $M_{\alpha_i}$ , no additional paths could have been introduced by the new transitions. Thus, the claim follows immediately from the induction hypothesis. In the second case, where  $q = q_0$  and  $q'$  is in  $M_{\alpha_i}$  a path must take the  $\varepsilon$ -transition to  $q_{0,i}$  and then take a path between  $q_{0,i}$  and  $q'$ . Since  $q_0$  is not marked, we have  $q_0 \xrightarrow{\varepsilon}_X q'$  iff  $q_{0,i} \xrightarrow{\varepsilon}_X q'$ , establishing the claim by induction hypothesis. The third case, where  $q$  is in  $M_{\alpha_i}$  and  $q' = q_f$  is analogous to the second one. Another case is  $q = q_0$  and  $q' = q_f$ . Since  $M_{\alpha_1}$  and  $M_{\alpha_2}$  are not connected, we have  $q \xrightarrow{\varepsilon}_X q'$  iff  $q_{0,1} \xrightarrow{\varepsilon}_X q_{f,1}$  or  $q_{0,2} \xrightarrow{\varepsilon}_X q_{f,2}$  with the same argument as used in cases two and three. Since no paths could be added from  $q_{0,i}$  to  $q_{f,i}$  when going over from  $M_{\alpha_i}$  to  $M_{\alpha}$ ,  $q_{0,i} \xrightarrow{\varepsilon}_X q_{f,i}$  holds for  $\xrightarrow{\varepsilon}_X$  constructed from  $M_{\alpha_i}$  iff it holds for  $\xrightarrow{\varepsilon}_X$  constructed from  $M_{\alpha}$ . Therefore we have  $\bigvee\{\bigwedge_{\psi_i \in X} v_i | q \xrightarrow{\varepsilon}_X q'\} \equiv \bigvee\{\bigwedge_{\psi_i \in X} v_i | q_{0,1} \xrightarrow{\varepsilon}_X q_{f,1}\} \vee \bigvee\{\bigwedge_{\psi_i \in X} v_i | q_{0,2} \xrightarrow{\varepsilon}_X q_{f,2}\} \equiv \varepsilon p(q_{0,1}, q_{f,1}, \alpha_1) \vee \varepsilon p(q_{0,2}, q_{f,2}, \alpha_2) = \varepsilon p(q, q', \alpha)$  using the induction hypothesis. The remaining cases include  $q = q_f$  with  $q' = q_0$  and  $q$  being in  $M_{\alpha_i}$  with  $q'$  being in  $M_{\alpha_{1-i}}$ . In both cases,  $q'$  is not reachable from  $q$  using only  $\varepsilon$ -transitions. Thus, the claim is established with similar arguments as in previous cases.

For the second claim, consider  $M_{\alpha^*}$ , where  $\varepsilon$ -transitions are added in between  $q_0$  and  $q_f$  compared to  $M_{\alpha}$ . We consider different cases how  $q'$  can be reached from  $q$  only taking  $\varepsilon$ -transitions. In the first case, where  $q, q' \in \{q_0, q_f\}$ , a single  $\varepsilon$ -transition without encountering state markings suffices and the claim is established as in previous cases. In the second case, we have  $q \in \{q_0, q_f\}$  and  $q'$  in  $M_{\alpha_i}$ . Since  $q_0$  and  $q_f$  are connected by an  $\varepsilon$ -transition, states can be reached from  $q_f$  the same way they can be reached from  $q_0$ . We argue that we only have to consider paths moving into  $M_{\alpha_i}$  once without leaving and reentering either

$M_{\alpha_i}$  or  $M_{\alpha_{1-i}}$  from the start. This is due to the fact that at the end of each of these paths, a subpath from  $q_0$  to  $q'$  has to be taken, making the set of encountered markings a superset of the encountered markings  $Y$  from a path only entering  $M_{\alpha_i}$  once. Since the conjunction over  $X$  is already accounted for by a conjunction over  $Y$  in the disjunction over paths, it can be eliminated from the disjunction resulting in an equivalent formula. The resulting disjunction is indeed exactly the same as for  $q_0, q'$  in the first case for which we have already shown that it is equivalent to  $\varepsilon p(q_0, q', \alpha)$ . This establishes the claim. In our third case, where  $q$  is in  $M_{\alpha_i}$  and  $q' \in \{q_0, q_f\}$ , we can argue analogously. The next case is where  $q$  and  $q'$  are both in  $M_{\alpha_i}$ . We argue that paths going through  $M_{\alpha_{1-i}}$  do not have to be accounted for with a similar argument as in case two: these paths have to leave  $M_{\alpha_i}$  at the start through  $q_f$  and reenter it at the end through  $q_0$  and visit a superset of state markings compared to a path that skips all subpaths through  $M_{\alpha_{1-i}}$ . We can use the same argument as in case two and the induction hypothesis to establish the claim. The last case is where  $q$  is in  $M_{\alpha_i}$  and  $q'$  is in  $M_{\alpha_{1-i}}$ . Compared to  $M_\alpha$ , where it was impossible to reach  $q'$  from  $q$ , we now have the ability to use newly introduced transitions to move from the final state of  $M_{\alpha_i}$  to the starting state of  $M_{\alpha_{1-i}}$ . We again do not have to consider paths taking this opportunity multiple times since they all encounter a superset of state markings compared to one of the paths we consider. Thus, the paths we consider encounter state markings on their way from  $q$  to  $q_{f,i}$  and on their way from  $q_{0,1-i}$  to  $q'$ . The claim can be established by using the induction hypothesis and previously used arguments.

**Case  $\alpha = \alpha_1 \alpha_2$ :** For the first claim, we consider three cases. In the first one,  $q$  and  $q'$  are both in  $M_{\alpha_i}$ . Since only transitions from  $M_{\alpha_1}$  into  $M_{\alpha_2}$  are possible but not backwards, a path inducing  $q \xrightarrow{\varepsilon}_X q'$  has to stay in  $M_{\alpha_i}$  the whole time. The claim then follows from the induction hypothesis. In the second case,  $q$  is in  $M_{\alpha_1}$  and  $q'$  is in  $M_{\alpha_2}$ . A path from  $q$  to  $q'$  has to transition through  $q_{f,1}$  and  $q_{0,2}$  to be able to switch automata, thus we have  $q \xrightarrow{\varepsilon}_X q'$  iff  $q \xrightarrow{\varepsilon}_Y q_{f,1}$  and  $q_{0,2} \xrightarrow{\varepsilon}_Z q'$  for some  $Y, Z$  with  $X = Y \cup Z$ . Since for state pairs inside one of the subautomata it does not matter whether  $\xrightarrow{\varepsilon}_X$  was constructed from  $M_\alpha$  or  $M_{\alpha_i}$ , the claim follows from the induction hypothesis. In the last case,  $q$  is in  $M_{\alpha_2}$  and  $q'$  is in  $M_{\alpha_1}$ . Since  $q'$  is not reachable from  $q$ ,  $q \xrightarrow{\varepsilon}_X q'$  can not hold for any  $X$  and the claim follows immediately.

The second claim requires a case discrimination with five cases. In the first one, we have  $q, q' \in \{q_{0,1}, q_{f,2}\}$ . Since both states are connected by  $\varepsilon$ -transitions and none of them is marked, the claim follows immediately from previously used arguments. In the second case, both  $q$  and  $q'$  are in  $M_{\alpha_1}$ . A path from  $q$  to  $q'$  can either stay in  $M_{\alpha_1}$  the whole time or move into and through  $M_{\alpha_2}$  by transitioning  $q_{f,1}, q_{0,2}$  and then  $q_{f,2}$  to reenter  $M_{\alpha_1}$  in  $q_{0,1}$ . Just like in previous cases, we do not have to consider paths where  $M_{\alpha_1}$  and  $M_{\alpha_2}$  are traversed multiple times. In addition to the arguments used in previous cases, we argue here that the behaviour of subpaths before leaving  $M_{\alpha_1}$  and after reentering  $M_{\alpha_1}$  combined are captured by  $\varepsilon p^*(q, q', \alpha_1)$ . Then the case follows from the induction hypothesis. The third case, where  $q$  and  $q'$  are both in  $M_{\alpha_2}$  is analogous to the previous



one. In the next case,  $q$  is in  $M_{\alpha_1}$  and  $q'$  is in  $M_{\alpha_2}$ . Since paths traversing the subautomata  $M_{\alpha_1}$  and  $M_{\alpha_2}$  multiple times do not have to be considered, we can argue just like for claim one in this case. The last case is the situation where  $q$  is in  $M_{\alpha_2}$  and  $q'$  is in  $M_{\alpha_1}$ . Again, paths traversing the subautomata multiple times do not have to be considered. The claim follows just like in the second case of claim one.

**Case  $\alpha = \alpha_1^*$ :** The first claim follows immediately from the second claim of the induction hypothesis. The second claim follows from the fact that the construction does not change when performing the  $*$  construction an additional time.

**Case  $\alpha = \psi_k$ :** For the first claim, we consider the different cases how  $q'$  can be  $\varepsilon$ -reached from  $q$  in  $M_\alpha$ . In the first case,  $q = q'$  with  $q \neq q_1$ , we have trivial reachability without encountering a state marking. Here, the claim is established as in previous cases. In the second claim, where  $q = q_i, q' = q_j$  with  $i > j$ ,  $q'$  is not reachable from  $q$  since the  $\varepsilon$ -transitions only point in the other direction. The claim is again established as in previous cases. In all other cases,  $q'$  can be reached from  $q$  with  $\varepsilon$ -transitions, but only with encountering the state marking in  $q_1$ . Since this is the only state marking in  $M_\alpha$ , we have  $\bigvee \{ \bigwedge_{\psi_i \in X} v_i | q \xrightarrow{\varepsilon}_X q' \} \equiv v_i = \varepsilon p(q, q', \alpha)$ , establishing the claim.

For the second claim,  $\varepsilon$ -transitions are added in between  $q_0$  and  $q_2$  when going over from  $M_\alpha$  to  $M_{\alpha^*}$ , making all states  $\varepsilon$ -reachable from all other states. For different states, the only difference is if the state marking in  $q_1$  can be avoided on the  $\varepsilon$ -path. This is exactly the case when neither  $q$  nor  $q'$  are  $q_1$ . When it can be avoided, we have  $\bigvee \{ \bigwedge_{\psi_i \in X} v_i | q \xrightarrow{\varepsilon}_X q' \} \equiv \bigwedge_{\psi_i \in \emptyset} v_i \vee \bigwedge_{\psi_i \in \{\psi_k\}} v_i \equiv \text{true} \vee \bigwedge_{\psi_i \in \{\psi_k\}} v_i \equiv \text{true}$  and the claim is established. In the other case, where it can not be avoided, we argue just as in the first claim.