Decidability and Expressivity of Ockhamist Propositional Dynamic Logics

Joseph Boudou, Emiliano Lorini

Decidability and Expressivity of Ockhamist Propositional Dynamic Logics

Joseph Boudou(✉) and Emiliano Lorini

IRIT-CNRS, Toulouse University, Toulouse, France
{joseph.boudou,lorini}@irit.fr

Abstract. Ockhamist Propositional Dynamic Logic (OPDL) is a logic unifying the family of dynamic logics and the family of branching-time temporal logics, two families of logic widely used in AI to model reactive systems and multi-agent systems (MAS). In this paper, we present two variants of this logic. These two logics share the same language and differ only in one semantic condition. The first logic embeds Bundled CTL* while the second embeds CTL*. We provide a 2EXPTIME decision procedure for the satisfiability problem of each variant. The decision procedure for the first variant of OPDL is based on the elimination of Hintikka sets while the decision procedure for the second variant relies on automata.

1 Introduction

In [2] a new logic, called Ockhamist Propositional Dynamic Logic (OPDL) has been introduced. This logic connects the family of dynamic logics with the family of branching-time temporal logics, two families of logic that are traditionally used in artificial intelligence for the verification of programs and for modelling autonomous agents and multi-agent systems (MAS). On the one hand, dynamic logics have been used to model actions of agents and their consequences as well as deontic notions such as obligation and permission. On the other hand, branching-time temporal logics have been used to model the evolution of the agents' attitudes and dispositions including beliefs, preferences and intentions as well as to specify communication protocols and to model dynamics of commitments in a multi-agent setting.

As shown in [2], OPDL offers the right "bridge" between these two families of logics, as it embeds in a *natural* and *polynomial* way both Propositional Dynamic Logic (PDL) [10] and Full Computation Tree Logic (CTL*) [14]. Existing embeddings of both PDL and CTL* are rather complicated and unnatural. For example, it is well-known that PDL and CTL* can be embedded in modal μ-calculus. However, although the embedding of PDL into modal μ-calculus is simple and direct, the embedding of CTL* into modal μ-calculus is rather complicated and doubly exponential in the length of the input formula [5]. Another logic that links PDL with CTL* is the extension of PDL with a repetition construct (PDL-Δ) by [16]. But again, the embedding of CTL* into PDL-Δ is rather complicated and doubly exponential in the length of the input formula [19].

OPDL can be conceived as the logic in the dynamic logic family based on the Ockhamist view of time. Ockhamist semantics for temporal logic have been widely studied [4,17,20]. The logic of agency STIT (the logic of "seeing to it that") by Belnap *et al.* [3] is based on such semantics. According to the Ockhamist conception of time (also called *indeterminist actualist*, see [20]) the truth of statements is evaluated with respect to a moment and to a particular *actual* linear history passing through that moment.[1]

The original semantics for OPDL given by [2] is based on the concept of OPDL Ockhamist model, which can be seen as an extension with a program component of Zanardo's Ockhamist model for branching-time temporal logics [20]. Specifically, in an OPDL Ockhamist model, temporal transitions between states are labelled with sets of atomic programs. A second variant of OPDL is studied by [2], called OPDL^{lts}. Like PDL, OPDL^{lts} is interpreted in labelled transition systems (LTS). However, while in PDL the truth of a formula is evaluated with respect to a state, in OPDL^{lts} it is evaluated with respect to a path.

The present paper furthers the study of OPDL by providing complexity results of the satisfiability problems of its different variants. Specifically, we introduce a new path semantics for OPDL, which allows for finer analyses of its different variants. The OPDL Ockhamist semantics is proved to correspond to the fusion closure condition in the path semantics. Observing that OPDL^{lts} studied by [2] lacks the conservative property, a new variant of OPDL, called OPDL^{lc}, is devised by adding the limit closure property to the path semantics, thereby imitating the difference between the semantics for Bundled CTL^* (BCTL^*) and the semantics for CTL^*. We show that the satisfiability problems of OPDL and OPDL^{lts} are both 2EXPTIME-complete, the same complexity as for CTL^*.

The rest of the paper is organized as follows. In the next section, the OPDL language and the Ockhamist semantics for OPDL are recalled from [2]. The path semantics framework is also introduced. Then, optimal decision procedures for the satisfiability of OPDL and OPDL^{lc} are presented in Sects. 3 and 4, respectively. We conclude in Sect. 5.[2]

2 Ockhamist Propositional Dynamic Logics

OPDL and OPDL^{lc} share the same language which is the language of PDL where one special atomic program $\equiv$ called the branching program is distinguished. Formally, assume a countable set $Prop = \{p, q, \ldots\}$ of atomic propositions and a countable set $Atm = \{a, b, \ldots\}$ of atomic programs (or actions). The language $\mathcal{L}_{\mathsf{OPDL}}(Prop, Atm)$ of OPDL consists of a set Prg of programs and a set Fml of formulas, defined as follows:

$$Prg: \ \alpha ::= a \mid \equiv \mid (\alpha_1; \alpha_2) \mid (\alpha_1 \cup \alpha_2) \mid \alpha^* \mid \varphi?$$
$$Fml: \varphi ::= p \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid [\![\alpha]\!]\varphi$$

[1] The Ockhamist view of branching time is traditionally opposed to the Peircean view [13,17]. According to the Peircean view, the truth of a temporal formula should be evaluated with respect either to some history or all histories starting in a given state.

[2] Due to space restriction, this version of the paper contains only sketches of proofs of some theorems.

where $\equiv$ is a syntactic symbol distinct from atomic programs. We adopt the standard definitions for the remaining Boolean operations. Implicit elimination of double negations is assumed: $\neg\neg\varphi$ is identified with φ. The dual $\langle\langle\alpha\rangle\rangle$ of the modality $[\![\alpha]\!]$ is defined by $\langle\langle\alpha\rangle\rangle\varphi \stackrel{\texttt{def}}{=} \neg[\![\alpha]\!]\neg\varphi$. We write $|\alpha|$ and $|\varphi|$ to denote the numbers of occurrences of symbols in the program α and the formula φ. Like for PDL, the formula $[\![\alpha]\!]\varphi$ has to be read as "φ holds after all possible executions of α".

2.1 Ockhamist Semantics

OPDL models are structures with two dimensions: a vertical dimension corresponding to the concept of history, a horizontal dimension corresponding to the concept of moment.

Definition 1. *An OPDL* model *is a tuple* $M = (W, \mathcal{Q}, \mathcal{L}, \mathcal{R}_{\equiv}, \mathcal{V})$ *where:*

- W *is a nonempty set of states (or worlds),*
- $\mathcal{Q}$ *is a partial function* $\mathcal{Q} : W \longrightarrow W$ *assigning a successor to states,*
- $\mathcal{L}$ *is a mapping* $\mathcal{L} : W \times W \longrightarrow 2^{Atm}$ *from pairs of states to sets of atomic programs such that* $\mathcal{L}(w, v) \neq \emptyset$ *iff* v *is the successor of* w*, i.e.,* $v = \mathcal{Q}(w)$*,*
- $\mathcal{R}_{\equiv} \subseteq W \times W$ *is an equivalence relation between states in* W*,*
- $\mathcal{V} : W \longrightarrow 2^{Prop}$ *is a valuation function for atomic propositions,*

and such that for all $w, v, u \in W$*:*

(C1) *if* $\mathcal{Q}(w) = v$ *and* $(v, u) \in \mathcal{R}_{\equiv}$ *then there is* $z \in W$ *such that* $(w, z) \in \mathcal{R}_{\equiv}$*,* $\mathcal{Q}(z) = u$ *and* $\mathcal{L}(z, u) = \mathcal{L}(w, v)$*.*
(C2) *if* $(w, v) \in \mathcal{R}_{\equiv}$ *then* $\mathcal{V}(w) = \mathcal{V}(v)$*.*

$\mathcal{R}_{\equiv}$-equivalence classes are called *moments*. A *history* starting in w_1 is a maximal sequence $\sigma = w_1, w_2, \ldots$ of states such that $w_{k+1} = \mathcal{Q}(w_k)$ for all positive k less than the length of σ.

Constraint C1 corresponds to what in Ockhamist semantics is called property of *weak diagram completion*. This means that if two worlds v and u are in the same moment and world w is a predecessor of v then, there exists a world z such that (i) w and z are in the same moment, (ii) u is the successor of z, (iii) the transition from w to v and the transition from z to u are labeled with the same set of action names. Constraint C2 just means that two worlds belonging to the same moment agree on the truth values of the atoms.

The truth of an OPDL formula is evaluated with respect to a world w in an OPDL model M.

Definition 2. *Let* $M = (W, \mathcal{Q}, \mathcal{L}, \mathcal{R}_{\equiv}, \mathcal{V})$ *be an OPDL model. Given a program* α*, we define a binary relation* $\mathcal{R}_\alpha$ *on* W *with* $(w, v) \in \mathcal{R}_\alpha$ *(or* $w\ \mathcal{R}_\alpha\ v$*) meaning that* v *is accessible from* w *by performing* α*. We also define a binary relation* $\models$

between worlds in M and formulas with $M, w \models \varphi$ meaning that formula φ is true at w in M. The rules inductively defining $\mathcal{R}_\alpha$ and $\models$ are:

$$\begin{aligned}
\mathcal{R}_a &= \{(w,v) \mid \mathcal{Q}(w) = v \text{ and } a \in \mathcal{L}(w,v)\} \\
\mathcal{R}_{\alpha_1;\alpha_2} &= \mathcal{R}_{\alpha_1} \circ \mathcal{R}_{\alpha_2} \\
\mathcal{R}_{\alpha_1 \cup \alpha_2} &= \mathcal{R}_{\alpha_1} \cup \mathcal{R}_{\alpha_2} \\
\mathcal{R}_{\alpha^*} &= (\mathcal{R}_\alpha)^* \\
\mathcal{R}_{\varphi?} &= \{(w,w) \mid M, w \models \varphi\}
\end{aligned}$$

and

$$\begin{aligned}
M, w \models p &\iff p \in \mathcal{V}(w); \\
M, w \models \neg\varphi &\iff M, w \not\models \varphi; \\
M, w \models \varphi \wedge \psi &\iff M, w \models \varphi \text{ and } M, w \models \psi; \\
M, w \models [\![\alpha]\!]\varphi &\iff \forall v \in W, \text{ if } w \,\mathcal{R}_\alpha\, v \text{ then } M, v \models \varphi.
\end{aligned}$$

An OPDL formula φ is OPDL valid, denoted by $\models_{\mathsf{OPDL}} \varphi$, iff for every OPDL model M and for every world w in M, we have $M, w \models \varphi$. An OPDL formula φ is OPDL satisfiable iff $\neg\varphi$ is not OPDL valid.

2.2 Path Semantics

In this section we describe the path semantics for $\mathcal{L}_{\mathsf{OPDL}}(Prop, Atm)$, inspired by the path semantics for branching time temporal logics [14]. In this semantics, the set of all histories is explicit in the model and formulas are interpreted over histories. We show that one variant of this semantics is equivalent to the Ockhamist semantics of the previous section, while another variant defines the OPDL^{lc} logic.

Notation. Given an alphabet Σ, Σ^* denotes the set of finite words over Σ, Σ^ω the set of infinite words and Σ^∞ the union of Σ^* and Σ^ω. Let $\sigma = w_1 w_2 \ldots$ be a finite or infinite word. The length of σ is denoted by $|\sigma|$. If σ is infinite then $|\sigma| = \omega$. For any $i \in 1..|\sigma|$, we use σ^i, $\sigma^{\leq i}$ and $\sigma^{\geq i}$ to denote respectively the i^{th} element w_i in σ, the prefix $w_1 \ldots w_i$ of σ up to its i^{th} element and the suffix $w_i w_{i+1} \ldots$ of σ from its i^{th} element. The notations $\sigma^{<i}$, $\sigma^{>i}$ and $\sigma^{i..j}$ are shorthands for $\sigma^{\leq i-1}$, $\sigma^{\geq i+1}$ and $(\sigma^{\leq j})^{\geq i}$, respectively.

Definition 3. *A* path model *is a tuple $M = (W, \mathcal{L}, B, \mathcal{V})$ where W is non-empty set of states, $\mathcal{L} : W \times W \longrightarrow 2^{Atm}$ is a function assigning a set of atomic programs to each pair of states, the bundle $B \subseteq W^\infty$ is a non-empty set of sequences of states (histories) such that for each sequence $\sigma = w_1, w_2, \ldots \in B$ and all $k \geq 1$ less than the length of σ, $\mathcal{L}(w_k, w_{k+1}) \neq \emptyset$ and $\mathcal{V} : W \longrightarrow 2^{Prop}$ is a valuation for the propositional variables. The binary relations $\mathcal{R}_\alpha$ over B*

for all programs α and the forcing relation $\models$ between M, sequences in B and formulas are defined by simultaneous induction such that:

$$\mathcal{R}_a = \{(\sigma_1, \sigma_2) \mid \sigma_2 = \sigma_1^{\geq 2} \text{ and } a \in \mathcal{L}(\sigma_1^1, \sigma_2^1)\}$$
$$\mathcal{R}_{\equiv} = \{(\sigma_1, \sigma_2) \mid \sigma_1^1 = \sigma_2^1\}$$

and

$$M, \sigma \models p \iff p \in \mathcal{V}(\sigma^1)$$
$$M, \sigma \models \neg\varphi \iff M, \sigma \not\models \varphi;$$
$$M, \sigma \models \varphi \wedge \psi \iff M, \sigma \models \varphi \text{ and } M, \sigma \models \psi;$$
$$M, \sigma \models [\![\alpha]\!]\varphi \iff \forall \sigma' \in B, \text{ if } \sigma \mathcal{R}_\alpha \sigma' \text{ then } M, \sigma' \models \varphi.$$

the missing cases being identical as to Definition 2.

The main interest in the path semantics is that, by adding additional conditions restricting the possible bundles, it gives a convenient framework to analyse and distinguish different logics based on the same language. We list some such conditions and discuss their impact on logics. We abusively write that a model has one of these conditions whenever its bundle has it.

Suffix closure. B is suffix closed iff for any sequence $\sigma \in B$ and any $k \in 1..|\sigma|$, $\sigma^{\geq k} \in B$. In contrast with CTL^*, as long as seriality is not imposed, this condition does not change the logic. But since this condition makes the definition of $\mathcal{R}_a$ more natural, we will assume path models have it.

Fusion closure. B is fusion closed iff for any two sequences $\sigma_1, \sigma_2 \in B$, if $\sigma_1^k = \sigma_2^{k'}$ for some k and k' then the sequence $\sigma_1^{<k}\sigma_2^{\geq k'}$ is in B. This condition corresponds to condition (C1). Indeed, we have the following theorem.

Theorem 1. *OPDL is the logic obtained by interpreting $\mathcal{L}_{\mathsf{OPDL}}(Prop, Atm)$ in the class of all suffix and fusion closed path models.*

Limit closure. B is limit closed iff whenever an infinite sequence $\sigma \in W^\omega$ is such that for all $k \geq 1$, there is a sequence $\sigma_k \in B$ such that $\sigma_k^{\leq k} = \sigma^{\leq k}$ then $\sigma \in B$. A similar condition makes the difference between BCTL^* and CTL^* [14]. The logic obtained by interpreting $\mathcal{L}_{\mathsf{OPDL}}(Prop, Atm)$ in the class of suffix, fusion and limit closed models is called OPDL^{lc}.

Seriality. B is serial iff all paths in B are infinite ($B \subseteq W^\omega$). Combining this condition with the suffix closure corresponds, in the Ockhamist semantics, to enforcing $\mathcal{Q}$ to be a total function. If Atm is infinite, then any path model satisfying a formula φ_0 can be turned into a serial path model satisfying φ_0 by choosing an atomic program e not occurring in φ_0 and by adding for each finite sequence $\sigma \in B$ a state w_σ such that w_σ is a successor by $\{e\}$ of itself and of the last state in σ. This transformation preserves satisfiability and the suffix closed, fusion closed and limit closed conditions. Therefore, since OPDL and OPDL^{lc} are conservative, we can assume that these logics are interpreted in serial path models.

Total seriality. B is totally serial iff B is the set of all infinite paths. By the constructions used in the proofs of Corollary 1 or Theorem 4, we can prove as a corollary of any of these theorems that the logic obtained by interpreting $\mathcal{L}_{\mathsf{OPDL}}(Prop, Atm)$ in the class of all suffix closed, fusion closed and totally serial models is OPDL^{lc}.

Total maximality. B is totally maximal iff B is the set of all maximal paths. In [2], the logic obtained by interpreting $\mathcal{L}_{\mathsf{OPDL}}(Prop, Atm)$ in the class of totally maximal models, called $\mathsf{OPDL}^{lts}(Prop, Atm)$, have been considered. But, in contrast with OPDL and OPDL^{lc}, $\mathsf{OPDL}^{lts}(Prop, Atm)$ is not conservative. We define a logic $L1$ in the language $\mathcal{L}(Prop, Atm)$ as being *conservative* iff every extensions $L2$ of $L1$ to the language $\mathcal{L}(Prop', Atm')$ where $Prop \subseteq Prop'$ and $Atm \subseteq Atm'$, is a conservative extension, i.e., the set of validities of $L2$ in the language $\mathcal{L}(Prop, Atm)$ is exactly the set of validities of $L1$. Intuitively, a logic is conservative if the validity of any formula is independent of the propositional variables and atomic program which does *not* occur in the formula. To prove that $\mathsf{OPDL}^{lts}(Prop, \{a\})$ is not conservative, consider the formula $[\![a]\!]\bot \wedge \langle\!\langle \equiv; a\rangle\!\rangle \top$. This formula is not $\mathsf{OPDL}^{lts}(Prop, \{a\})$ satisfiable but is $\mathsf{OPDL}^{lts}(Prop, \{a, b\})$ satisfiable. In the present work, we will study OPDL^{lc} (which is conservative) instead of $\mathsf{OPDL}^{lts}(Prop, Atm)$. It can easily be proved that if Atm is infinite then OPDL^{lc} and $\mathsf{OPDL}^{lts}(Prop, Atm)$ are the same logic. Moreover, the proof from [2] that CTL^* can be embedded into OPDL^{lts} can easily be adapted to prove that CTL^* can be embedded into OPDL^{lc}.

3 Optimal Decision Procedure for OPDL

We describe a decision procedure for the satisfiability problem of OPDL, based on the *elimination of Hintikka sets* procedure devised for PDL by Pratt [12] and adapted to BCTL^* by Reynolds [15]. The general idea is to construct a syntactic structure which contains all the possible states then to eliminate the states preventing the structure to be a model. For PDL the possible states are Hintikka sets (*hues* in [15]). For BCTL^*, states are sets of Hintikka sets, called *clusters* in this paper (*colors* in [15]). For OPDL, states must be clusters too, but because of formulas like $\langle\!\langle a\rangle\!\rangle p \wedge [\![b]\!]\neg p \wedge \langle\!\langle \equiv\rangle\!\rangle \langle\!\langle b\rangle\!\rangle p$, the atomic programs labeling edges have to be considered. Hence the syntactic structures are more involved than for PDL or BCTL^*. We study these syntactic structures before introducing the decision procedure for OPDL. Properties of syntactic structures are used for the automata-based procedure of Sect. 4 too.

3.1 Syntactic Structures

Given a formula φ_0, the Fischer-Ladner closure $FL(\varphi_0)$ of φ_0 is defined as for PDL (see [9] for details) except that we enforce $FL(\varphi_0)$ to be closed under negation: $\psi \in FL(\varphi_0)$ iff $\neg\psi \in FL(\varphi_0)$. Since implicit elimination of double negation is assumed, the well-known result that the cardinal of $FL(\varphi_0)$ is linear in $|\varphi_0|$ remains. We write $SP(\varphi_0)$ to denote the set $\{\alpha \mid \exists\varphi,\ \langle\!\langle\alpha\rangle\!\rangle\varphi \in FL(\varphi_0)\}$.

Definition 4. *A set $\mathcal{H} \subset FL(\varphi_0)$ is a* Hintikka set *for φ_0 iff all the following conditions are satisfied:*

- *for any $\neg\varphi \in FL(\varphi_0)$, $\varphi \in \mathcal{H}$ iff $\neg\varphi \notin \mathcal{H}$*
- *for any $\varphi \wedge \psi \in FL(\varphi_0)$, $\varphi \wedge \psi \in \mathcal{H}$ iff $\varphi \in \mathcal{H}$ and $\psi \in \mathcal{H}$*
- *for any $[\![\alpha;\beta]\!]\varphi \in FL(\varphi_0)$, $[\![\alpha;\beta]\!]\varphi \in \mathcal{H}$ iff $[\![\alpha]\!][\![\beta]\!]\varphi \in \mathcal{H}$*
- *for any $[\![\alpha \cup \beta]\!]\varphi \in FL(\varphi_0)$, $[\![\alpha \cup \beta]\!]\varphi \in \mathcal{H}$ iff $[\![\alpha]\!]\varphi \in \mathcal{H}$ and $[\![\beta]\!]\varphi \in \mathcal{H}$*
- *for any $[\![\alpha^*]\!]\varphi \in FL(\varphi_0)$, $[\![\alpha^*]\!]\varphi \in \mathcal{H}$ iff $\varphi \in \mathcal{H}$ and $[\![\alpha]\!][\![\alpha^*]\!]\varphi \in \mathcal{H}$*
- *for any $[\![\varphi?]\!]\psi \in FL(\varphi_0)$, $[\![\varphi?]\!]\psi \in \mathcal{H}$ iff $\neg\varphi \in \mathcal{H}$ or $\psi \in \mathcal{H}$*
- *if $[\![\equiv]\!]\varphi \in \mathcal{H}$ then $\varphi \in \mathcal{H}$*

Definition 5. *A set $\mathcal{C}$ of Hintikka sets for φ_0 is a* cluster *for φ_0 iff $\mathcal{C} \neq \emptyset$ and for any $\mathcal{H}_1, \mathcal{H}_2 \in \mathcal{C}$ the following conditions are satisfied:*

- *for any propositional variable $p \in FL(\varphi_0)$, $p \in \mathcal{H}_1$ iff $p \in \mathcal{H}_2$*
- *for any formula $[\![\equiv]\!]\varphi \in FL(\varphi_0)$, $[\![\equiv]\!]\varphi \in \mathcal{H}_1$ iff $[\![\equiv]\!]\varphi \in \mathcal{H}_2$*

Given a set $P \subseteq Atm$ of atomic programs, the successor relation S_P over Hintikka sets is defined such that $\mathcal{H}_1\ S_P\ \mathcal{H}_2$ iff (i) for any formula $\langle\!\langle a \rangle\!\rangle\varphi \in \mathcal{H}_1$, $a \in P$ and (ii) for any formula $\langle\!\langle a \rangle\!\rangle\varphi \in FL(\varphi_0)$ such that $a \in P$, $\langle\!\langle a \rangle\!\rangle\varphi \in \mathcal{H}_1$ iff $\varphi \in \mathcal{H}_2$. This relation is extended to clusters: $\mathcal{C}_1\ S_P\ \mathcal{C}_2$ iff for all $\mathcal{H}_2 \in \mathcal{C}_2$ there exists $\mathcal{H}_1 \in \mathcal{C}_1$ such that $\mathcal{H}_1\ S_P\ \mathcal{H}_2$.

A syntactic structure is a pseudo-model where the valuation has been replaced with a function assigning clusters and where the bundle is implicit. Intuitively, each Hintikka set in the cluster associated to a state w corresponds to the set of formulas satisfied by a history starting at w.

Definition 6. *A* syntactic structure *for a formula φ_0 is a tuple $\mathcal{S} = (W, \mathcal{L}, \mathfrak{C})$ where W is a non-empty set of states, $\mathcal{L}$ assigns a set of atomic programs to each pair of states, $\mathfrak{C}$ assigns a cluster for φ_0 to each state such that for all $w, x \in W$, if $\mathcal{L}(w,x) \neq \emptyset$ then $\mathfrak{C}(w)\ S_{\mathcal{L}(w,x)}\ \mathfrak{C}(x)$. A syntactic structure is* standard *iff (i) $\varphi_0 \in \mathcal{H}$ for some $\mathcal{H} \in \mathfrak{C}(w)$ and some $w \in W$ and (ii) for all $w \in W$, there exists $x \in W$ such that $\mathcal{L}(w,x) \neq \emptyset$.*

A *path* in a syntactic structure $\mathcal{S}$ is a (possibly infinite) non-empty sequence π over the alphabet composed by the special *branching* symbol $\bullet$ and all the couples $(\mathcal{H}, w)$ where $w \in W$ and $\mathcal{H} \in \mathfrak{C}(w)$. Any path π must satisfy all the following conditions, for all $k \in 1..|\pi|$:

- $\pi^1 \neq \bullet$ and if $|\pi| < \omega$, $\pi^{|\pi|} \neq \bullet$;
- if $\pi^k = \bullet$ then $\pi^{k-1} = (\mathcal{H}, w)$ and $\pi^{k+1}(\mathcal{H}', w)$ for some $w \in W$ and some $\mathcal{H}, \mathcal{H}' \in \mathfrak{C}(w)$;
- if $\pi^k = (\mathcal{H}_k, w_k)$ and $\pi^{k+1} = (\mathcal{H}_{k+1}, w_{k+1})$ then $\mathcal{L}(w_k, w_{k+1}) \neq \emptyset$ and $\mathcal{H}_k\ S_{\mathcal{L}(w_k, w_{k+1})}\ \mathcal{H}_{k+1}$.

Intuitively, a finite path π corresponds to a possible execution of some programs (different programs may have some common possible executions). When this is the case, we say that the path *carries* the program. This relation between a

finite path and a program is defined formally as the least relation satisfying the following conditions:

- $(\mathcal{H}_1, w_1)(\mathcal{H}_2, w_2)$ carries a iff $a \in \mathcal{L}(w_1, w_2)$.
- $(\mathcal{H}_1, w) \bullet (\mathcal{H}_2, w)$ carries $\equiv$.
- $(\mathcal{H}_1, w_1)$ carries $\varphi?$ iff $\varphi \in \mathcal{H}_1$.
- π carries $(\alpha \cup \beta)$ iff π carries α or β.
- π carries $(\alpha; \beta)$ iff for some $m \in 1..|\pi|$, $\pi^{\leq m}$ carries α and $\pi^{\geq m}$ carries β.
- π carries α^* iff there is a non-empty list $k_0, \ldots, k_m$ such that $k_0 = 1$, $k_m = |\pi|$ and for all $i < m$, $k_i < k_{i+1}$ and $\pi^{k_i \ldots k_{i+1}}$ carries α.

An *unbranching path* is a path which contains no occurrences of the branching symbol $\bullet$. The *trunk* of a path is its longest unbranching prefix. The *support* of an unbranching path $(\mathcal{H}_1, w_1)(\mathcal{H}_2, w_2)\ldots$ is the sequence $w_1 w_2 \ldots$.

An *eventuality chain* is a non-empty sequence $\eta = \alpha_1 \ldots \alpha_n \varphi$ where the last element is a formula and the other elements are programs. To an eventuality chain $\eta = \alpha_1 \ldots \alpha_n \varphi$ corresponds the formula $\mathrm{form}\,(\eta) = \langle\!\langle \alpha_1 \rangle\!\rangle \ldots \langle\!\langle \alpha_n \rangle\!\rangle \varphi$. This correspondence is not injective, for instance the eventuality chains aap, $a\langle\!\langle a \rangle\!\rangle p$ and $\langle\!\langle a \rangle\!\rangle \langle\!\langle a \rangle\!\rangle p$ all correspond to the same formula $\langle\!\langle a \rangle\!\rangle \langle\!\langle a \rangle\!\rangle p$. The *maximal eventuality chain* for a formula φ is the longest eventuality chain η such that $\mathrm{form}\,(\eta) = \varphi$. *Fulfillment* of an eventuality chain η by a path π is defined inductively as follows:

- The path π fulfills a one-element eventuality chain $\eta = \varphi$ iff $\pi = (\mathcal{H}_1, w_1)$ and $\varphi \in \mathcal{H}_1$ for some state w_1 and some Hintikka set $\mathcal{H}_1 \in \mathfrak{C}(w_1)$;
- The path π fulfills an eventuality chain $\eta = \alpha\eta'$ iff there is $k \in 1..|\pi|$ such that $\pi^{\leq k}$ carries α and $\pi^{\geq k}$ fulfills η'.

For any eventuality chain $\eta = \alpha\varphi$ of length two, the corresponding formula $\langle\!\langle \alpha \rangle\!\rangle \varphi$ is called an *eventuality* and any path fulfilling η is said to *fulfill the eventuality* $\langle\!\langle \alpha \rangle\!\rangle \varphi$. A state $w \in W$ is fulfilling if for any Hintikka set $\mathcal{H} \in \mathfrak{C}(w)$ and any eventuality $\langle\!\langle \alpha \rangle\!\rangle \varphi \in \mathcal{H}$, there is a path π from $(\mathcal{H}, w)$ fulfilling $\langle\!\langle \alpha \rangle\!\rangle \varphi$. A syntactic structure $\mathcal{S}$ *fulfills all eventualities* iff all its states are fulfilling. A *justifying path* is an infinite unbranching path π such that for all k, if $\pi_k = (\mathcal{H}_k, w_k)$ for some $\mathcal{H}_k$ and w_k then for any eventuality $\langle\!\langle \alpha \rangle\!\rangle \varphi \in \mathcal{H}_k$, there is a fulfilling path π' for $\langle\!\langle \alpha \rangle\!\rangle \varphi$ starting at $(\mathcal{H}_k, w_k)$ such that the trunk of π' is a prefix of $\pi^{\geq k}$.

We can now state the main result of this section.

Theorem 2. *A formula φ_0 is OPDL satisfiable if and only if there is a standard syntactic structure for φ_0 fulfilling all eventualities.*

Proof (Proof sketch). We only detail the right-to-left direction. Given a standard syntactic structure $\mathcal{S} = (W, \mathcal{L}, \mathfrak{C})$ for φ_0 fulfilling all eventualities, we define the path model $M = (W, \mathcal{L}, B, \mathcal{V})$ such that B is the set of supports of the justifying paths in $\mathcal{S}$ and $\mathcal{V}(w) = \mathcal{H} \cap \mathit{Prop}$ for any $\mathcal{H} \in \mathfrak{C}(w)$. Two steps are difficult in proving that M is an OPDL path model satisfying φ_0: the proof that B is fusion-closed and the proof of the following Existence Lemma.

Lemma 1 (Existence Lemma). *For any finite unbranching path π in a standard syntactic structure $\mathcal{S}$ fulfilling all eventualities, there is a justifying path π' in $\mathcal{S}$ such that π is a prefix of π'.*

For BCTL^*, these two points are resolved by the fact that any eventuality $\varphi\ \mathcal{U}\ \psi$ is either resolved at the current state or still satisfied in the successor state. For OPDL, we need the Witness Lemma below. To state this lemma, we inductively define the function next from eventuality chains to sets of pairs composed of a set of formulas (the guard) and an eventuality chain:

$$\begin{aligned}
\mathrm{next}(\varphi) &= \{(\emptyset, \varphi)\} & \mathrm{next}(\psi?\eta) &= \{(G \cup \{\psi\}, \eta') \mid (G, \eta') \in \mathrm{next}(\eta)\} \\
\mathrm{next}(a\eta) &= \{(\emptyset, a\eta)\} & \mathrm{next}((\beta_1 \cup \beta_2)\eta) &= \mathrm{next}(\beta_1\eta) \cup \mathrm{next}(\beta_2\eta) \\
\mathrm{next}(\equiv\eta) &= \{(\emptyset, \equiv\eta)\} & \mathrm{next}((\beta_1; \beta_2)\eta) &= \mathrm{next}(\beta_1\beta_2\eta) \\
\mathrm{next}(\alpha^*\eta) &= \mathrm{next}(\eta) \cup \{(G, \beta_1 \ldots \beta_{n'-1}\alpha^*\eta) \mid n' > 1 \text{ and} \\
& & & (G, \beta_1 \ldots \beta_{n'-1}\mathrm{form}\,(\alpha^*\eta)) \in \mathrm{next}(\alpha\mathrm{form}\,(\alpha^*\eta))\}
\end{aligned}$$

Lemma 2 (Witness Lemma). *For any syntactic structure $\mathcal{S} = (W, \mathcal{L}, \mathfrak{C})$, any state $w \in W$, any Hintikka set $\mathcal{H} \in \mathfrak{C}(w)$, any eventuality chain η_1 such that $form\,(\eta_1) \in \mathcal{H}$ and any path π in $\mathcal{S}$ from $(\mathcal{H}, w)$, π fulfills η_1 if and only if there is $(G, \eta_2) \in next(\eta_1)$ such that $G \cup \{form\,(\eta_2)\} \subseteq \mathcal{H}$ and π fulfills η_2.*

The proof of the Witness Lemma is by induction on the sum $\sum_{k=1}^{|\eta_1|-1} \left|\eta_1^k\right|$ of the length of the programs in η. □

In the proof of Theorem 2, we construct from a standard syntactic structure $\mathcal{S} = (W, \mathcal{L}, \mathfrak{C})$ for φ_0 the path model $M = (W, \mathcal{L}, B, \mathcal{V})$ in which B is the set of supports of the justifying paths in $\mathcal{S}$. Therefore if the set of the supports of the justifying paths in $\mathcal{S}$ is limit closed then B is limit closed too. Hence the following corollary can be deduced from Theorem 2.

Corollary 1. *A formula φ_0 is OPDL^{lc} satisfiable if and only if there is a standard syntactic structure $\mathcal{S}$ for φ_0 which fulfills all eventualities and such that the set of the supports of the justifying paths in $\mathcal{S}$ is limit closed.*

3.2 The Optimal Decision Procedure

We describe a procedure which, given a formula φ_0, either fails or exhibits a standard syntactic structure for φ_0 fulfilling all eventualities. The procedure inductively constructs a finite sequence $\mathcal{S}_0, \ldots, \mathcal{S}_n$ of syntactic structures for φ_0. The initial syntactic structure $\mathcal{S}_0 = (W_0, \mathcal{L}_0, \mathfrak{C}_0)$ is defined such that:

- W_0 is the set of all pairs $(P, \mathcal{C})$ where P is a non-empty subset of $SP\,(\varphi_0) \cup \{e\}$ for some fixed $e \notin SP\,(\varphi_0)$ and $\mathcal{C}$ is a cluster for φ_0,
- $\mathcal{L}((P_1, \mathcal{C}_1), (P_2, \mathcal{C}_2)) = P_2$ if $\mathcal{C}_1\ S_{P_2}\ \mathcal{C}_2$ and is empty otherwise,
- $\mathfrak{C}(P, \mathcal{C}) = \mathcal{C}$.

Then for all k, the syntactic structure $\mathcal{S}_{k+1}$ is constructed from $\mathcal{S}_k = (W_k, \mathcal{L}_k, \mathfrak{C}_k)$ by removing from W_k the states $(P, \mathcal{C})$ which are not fulfilling or such that for some $\mathcal{H} \in \mathcal{C}$, there is no $(P', \mathcal{C}') \in W_k$ and $\mathcal{H}' \in \mathcal{C}'$ such that $\mathcal{C}\ S_{P'}\ \mathcal{C}'$ and $\mathcal{H}\ S_{P'}\ \mathcal{H}'$.

There exists a constant C such that the number of states in W_0 for any φ_0 is bounded by $2^{2^{C \cdot \ell}}$ where $\ell = |\varphi_0|$. Therefore, for some $n \leq 2^{2^{C \cdot \ell}}$ no state can be eliminated from $\mathcal{S}_n$. The procedure terminates successfully iff there is a state $(P, \mathcal{C}) \in W_n$ and a Hintikka set $\mathcal{H} \in \mathcal{C}$ such that $\varphi_0 \in \mathcal{H}$. By Theorem 2, the decision procedure is sound and complete. Since the satisfiability problem of OPDL is 2EXPTIME-hard [2], we have the following theorem.

Theorem 3. *The satisfiability problem of OPDL is 2EXPTIME-complete.*

4 Optimal Decision Procedure for OPDL^{lc}

The procedure of the Sect. 3 is difficult to adapt to OPDL^{lc} because no simple condition can be checked during the construction of the syntactic structure to guarantee that the set of the supports of all justifying paths is limit closed. Therefore, we first prove that OPDL^{lc} has a particular tree model property. Then we use this property to reduce the satisfiability problem of OPDL^{lc} to the (dual of) the emptiness problem of an automaton on infinite trees. Because syntactic structures are more convenient than models for decision procedures, we prove a *tree syntactic structure property*, from which the usual tree model property can be deduced using the construction of Sect. 3.1.

4.1 Tree Model Property of OPDL^{lc}

An N-ary ω-tree over an alphabet Σ is a function $T : [1..N]^* \longrightarrow \Sigma$. In such a tree, nodes are labeled with elements of Σ. A *branch* in T is an infinite sequence $\sigma_1 = \lambda_1 \lambda_2 \ldots$ for which there exists $\sigma_2 \in [1..N]^\omega$ and $i \in \mathbb{N}$ such that for all $k > 0$, $\lambda_k = \sigma_2^{\leq i+k}$. Like in the previous section, we need nodes to be labeled with pairs $(P, \mathcal{C})$ where P is the set of atomic programs labeling the incoming edge and $\mathcal{C}$ is a cluster. To simulate incomplete trees, we allow P to be empty, in which case the branch is said to be pruned.

Definition 7. *An N-ary syntactic tree for a formula φ_0 is an N-ary ω-tree T over $\Sigma = 2^{Atm} \times Clusters(\varphi_0)$ where $Clusters(\varphi_0)$ is the set of clusters on φ_0 and such that:*

1. $T_P(\epsilon) = \emptyset$ and there is $\sigma \in [1..N]^\omega$ such that for all $i > 0$, $T_P(\sigma^{\leq i}) \neq \emptyset$;
2. for all $\lambda \in [1..N]^*$ and $k \in 1..N$, $T_P(\lambda k) = \emptyset$ or $T_C(\lambda)\ S_{T_P(\lambda k)}\ T_C(\lambda k)$.

where T_P and T_C are the projections of T on 2^{Atm} and $Clusters(\varphi_0)$, respectively. A branch σ in T is valid *if for all $k > 1$, $T_P(\sigma^k) \neq \emptyset$ and* pruned *otherwise.*

To any N-ary syntactic tree $T = (T_P, T_C)$ naturally corresponds the syntactic structure $\mathcal{S}(T) = ([1..N]^*, \mathcal{L}, T_C)$ where $\mathcal{L}(\lambda_1, \lambda_2) = T_P(\lambda_2)$ if $\lambda_2 = \lambda_1 k$ for some $k \in 1..N$ and is the empty set otherwise. Therefore, an N-ary syntactic tree can be seen as a tree syntactic structure. Indeed, we will abusively write about paths in syntactic trees. For the following definition of a good syntactic tree, since we do not assume that the corresponding syntactic structure fulfills all eventualities, we adapt the definition of a justifying path. A *pseudo-justifying path* is an infinite unbranching path π such that for all $k > 0$, if $\pi^k = (\mathcal{H}_k, w_k)$ then for any eventuality $\langle\langle\alpha\rangle\rangle\varphi \in \mathcal{H}_k$ there is $\ell \geq k$ such that $\pi^\ell = (\mathcal{H}_\ell, w_\ell)$ and either $\pi^{k..\ell}$ fulfills $\langle\langle\alpha\rangle\rangle\varphi$ or there is an eventuality chain η such that $\eta^1 = \equiv$, form $(\eta) \in \mathcal{H}_\ell$ and for any path π_2 from π^ℓ fulfilling η, $\pi^{k..(\ell-1)}\pi_2$ fulfills $\langle\langle\alpha\rangle\rangle\varphi$. By the Witness Lemma, any justifying path is a pseudo-justifying path.

Definition 8. *An N-ary syntactic tree $T = (T_P, T_C)$ for a formula φ_0 is* good *iff all the following conditions hold:*

1. *any valid branch σ is the support of a pseudo-justifying path;*
2. *for any node λ in T, if $T_P(\lambda) \neq \emptyset$ and there is $\mathcal{H} \in T_C(\lambda)$ such that $\langle\langle\equiv\rangle\rangle\psi \in \mathcal{H}$ for some formula ψ, then there is a finite path π in T from $(\mathcal{H}', \lambda)$ fulfilling the maximal eventuality chain for ψ;*
3. *there is a pseudo-justifying path in T from $(\mathcal{H}, \epsilon)$ such that $\varphi_0 \in \mathcal{H}$.*

Let $N^{\equiv}_{\varphi_0}$ be the number of eventualities of the form $\langle\langle\equiv\rangle\rangle\psi$ in $FL(\varphi_0)$ plus one. The tree property of OPDL^{lc} is stated as follows.

Theorem 4. *A formula φ_0 is OPDL^{lc} satisfiable iff there is a good $N^{\equiv}_{\varphi_0}$-ary syntactic tree for φ_0.*

Proof (Proof sketch). We only detail the construction for the left-to-right direction, which is inspired by a similar construction for CTL^* [7]. Suppose φ_0 is satisfiable. By Corollary 1, there is a standard syntactic structure $\mathcal{S} = (W, \mathcal{L}, \mathfrak{C})$ for φ_0 which fulfills all eventualities and such that the set of the supports of the justifying paths in $\mathcal{S}$ is limit closed. Let $\langle\langle\equiv\rangle\rangle\psi_2, \ldots, \langle\langle\equiv\rangle\rangle\psi_{N^{\equiv}_{\varphi_0}}$ be an ordering of the eventualities of the form $\langle\langle\equiv\rangle\rangle\psi$ in $FL(\varphi_0)$. We first define the $N^{\equiv}_{\varphi_0}$-ary ω-tree T_{path} over the alphabet of all the paths in $\mathcal{S}$ plus the empty word ϵ. By Lemma 1, there is a justifying path π_0 from $(\mathcal{H}_0, w_0)$. We label the root of T_{path} with this path: $T_{\text{path}}(\epsilon) = \pi_0$. For each node $\lambda \in [1..N^{\equiv}_{\varphi_0}]^*$, if $T_{\text{path}}(\lambda) \neq \epsilon$, the labeling path continues with the first successor: $T_{\text{path}}(\lambda 1) = T_{\text{path}}(\lambda)^{\geq 2}$. For the other successors $k \in 2..N^{\equiv}_{\varphi_0}$ of λ, let $(\mathcal{H}_\lambda, w_\lambda) = T_{\text{path}}(\lambda)^1$. If $\langle\langle\equiv\rangle\rangle\psi_k \in \mathcal{H}_\lambda$ then let π_1 be the shortest path fulfilling the maximal eventuality chain for ψ_k and such that $\pi_1^1 = (\mathcal{H}', w_\lambda)$ for some $\mathcal{H}'$. By Lemma 1, there is a justifying path $\pi_{\lambda k}$ whose prefix is the trunk of π_1. We label the k^{th} successor of λ with it: $T_{\text{path}}(\lambda k) = \pi^{\geq 2}_{\lambda k}$. Otherwise, if $\langle\langle\equiv\rangle\rangle\psi_{k-1} \notin \mathcal{H}_\lambda$ then $T_{\text{path}}(\lambda k) = \epsilon$. All successors of a node labeled with ϵ are labeled with ϵ. Finally, the good $N^{\equiv}_{\varphi_0}$-ary syntactic tree T for φ_0 is constructed from T_{path} as follows. For the root node, $T(\epsilon) = (\emptyset, \mathfrak{C}(w_0))$. For $\lambda \in [1..N^{\equiv}_{\varphi_0}]^*$ and $k \in 1..N^{\equiv}_{\varphi_0}$, if $T_{\text{path}}(\lambda)^1 = (\mathcal{H}_\lambda, w_\lambda)$ and $T_{\text{path}}(\lambda k)^1 = (\mathcal{H}_{\lambda k}, w_{\lambda k})$ then $T(\lambda k) = (\mathcal{L}(w_\lambda, w_{\lambda k}), \mathfrak{C}(w_{\lambda k}))$. Otherwise, $T(\lambda k) = (\emptyset, \mathcal{C})$ for some arbitrary cluster $\mathcal{C}$. □

4.2 Automata-based Decision Procedure for OPDL^{lc}

By Theorem 4, whenever a formula φ_0 is satisfiable, there is a good $N^{\equiv}_{\varphi_0}$-ary syntactic tree for φ_0. Therefore, we construct an automaton which recognizes exactly the good $N^{\equiv}_{\varphi_0}$-ary syntactic trees for φ_0. We first recall the definitions of the automata used in the procedure before describing the construction of our automaton.

A Büchi word automaton is a tuple $\mathcal{A} = (\Sigma, S, \rho, S_0, F)$ where Σ is the input alphabet, S is the set of states of the automaton, $\rho : S \times \Sigma \longrightarrow 2^S$ is a non-deterministic transition function, $S_0 \subseteq S$ is the set of initial states and $F \subseteq S$ is the termination condition. Given an infinite word μ over Σ, a *run* of $\mathcal{A}$ on μ is a word r over S such that $r^1 \in S_0$ and for all $k \geq 1$, $r^{k+1} \in \rho(r^k, \mu^k)$. The set of states occurring infinitely often in a run r is denoted by $\inf(r)$. A word μ is accepted by $\mathcal{A}$ iff there is a run r of $\mathcal{A}$ on μ such that $\inf(r) \cap F \neq \emptyset$. By extension, a Büchi word automaton accepts a tree iff it accepts all its branches seen as words over the labels of the trees's nodes.

A Street tree automaton is a tuple $\mathcal{A} = (\Sigma, S, \rho, S_0, F)$ similar to a Büchi word automaton except that $\rho : S \times \Sigma \longrightarrow 2^{S^N}$ assigns a set of N-ary tuples of states and $F \subseteq 2^S \times 2^S$ is a set of pairs of set of states. Given an N-ary ω-tree T over Σ, a run of $\mathcal{A}$ on T is a tree T_r over S such that $T_r(\epsilon) \in S_0$ and for all $\lambda \in [1..N]^*$, $(T_r(\lambda 1), \ldots, T_r(\lambda N)) \in \rho(T_r(\lambda), T(\lambda))$. For all branch σ in T_r, the set of states occurring infinitely often in σ is denoted by $\inf(\sigma)$. A tree T is accepted by $\mathcal{A}$ iff there is a run T_r of $\mathcal{A}$ on T such that for any branch σ in T_r and any pair $(A, B) \in F$, if $\inf(\sigma) \cap A \neq \emptyset$ then $\inf(\sigma) \cap B \neq \emptyset$.

Given a formula φ_0 we devise a Streett tree automaton $\mathcal{A}$ which recognizes exactly the good $N^{\equiv}_{\varphi_0}$-ary syntactic trees for φ_0. We first describe three automata, each checking conditions from Definitions 7 and 8. Let $\Sigma = 2^{Atm} \times \text{Clusters}(\varphi_0)$.

Condition (2) of Definition 7 is checked by the "successor" Büchi word automaton $\mathcal{A}_S = (\Sigma, S_S, \rho_S, S_{S,0}, F)$ where S_S is the set of clusters on φ_0 plus the special state I, $S_{S,0} = \{I\}$, $F_S = S_S$ and $s_1 \in \rho_S(s_0, (P, \mathcal{C}))$ iff (i) $s_1 = \mathcal{C}$ and (ii) $P = \emptyset$ or $s_0\ S_P\ s_1$.

Condition (1) of Definition 8 is checked by the "justifying" Büchi word automaton $\mathcal{A}_J = (\Sigma, S_J, \rho_J, S_{J,0}, F_J)$ where

- S_J is the set of pairs $(\mathcal{H}, E)$ where E is a set of eventuality chains to be fulfilled and $\mathcal{H}$ is either a Hintikka set of the parent cluster or the empty set if the current node is the root or $FL(\varphi_0)$ if the current branch is pruned;
- $S_{J,0} = \{(\emptyset, \emptyset)\}$ and $F_J = \{(\mathcal{H}, E) \in S_J \mid \mathcal{H} \neq \emptyset \text{ and } E = \emptyset\}$;
- $(\mathcal{H}_1, E_1) \in \rho_J((\mathcal{H}_0, E_0), (P, \mathcal{C}))$ if one of the following condition holds:
  - $\mathcal{H}_0$ is a Hintikka set, $E_0 \neq \emptyset$, $\mathcal{H}_1 \in \mathcal{C}$, $P \neq \emptyset$, $\mathcal{H}_0\ S_P\ \mathcal{H}_1$ and for all $\eta_0 \in E_0$, $\text{form}(\eta_0) \in \mathcal{H}_1$ and there is $(G_1, \eta_1) \in \text{next}(\eta_0)$ such that $G_1 \cup \{\text{form}(\eta_1)\} \subseteq \mathcal{H}_1$ and if $\eta_1^1 \in Atm$ then $\eta_1^{\geq 2} \in E_1$.
  - $\mathcal{H}_0 \neq FL(\varphi_0)$, $E_0 = \emptyset$, $\mathcal{H}_1 \in \mathcal{C}$, if $\mathcal{H}_0 \neq \emptyset$ then $P \neq \emptyset$ and $\mathcal{H}_0\ S_P\ \mathcal{H}_1$ and for any eventuality $\langle\langle\alpha\rangle\rangle\varphi \in \mathcal{H}_1$, there is $(G_1, \eta_1) \in \text{next}(\alpha\varphi)$ such that $G_1 \cup \{\text{form}(\eta_1)\} \subseteq \mathcal{H}_1$ and if $\eta_1^1 \in Atm$ then $\eta_1^{\geq 2} \in E_1$.
  - $\mathcal{H}_1 = FL(\varphi_0)$ and $E_1 \neq \emptyset$.
  - $\mathcal{H}_1 = FL(\varphi_0)$, $E_1 = \emptyset$ and either $E_0 = \emptyset$ or $\mathcal{H}_0 \neq \emptyset$ and $P = \emptyset$.

Finally, the "existential" Büchi tree automaton $\mathcal{A}_E = (\Sigma, S_E, \rho_E, S_{E,0}, F_E)$ ensures that there is a pseudo-justifying path π from $(\mathcal{H}_1, \epsilon)$ where $\varphi_0 \in \mathcal{H}_1$ and such that the support of π is the branch obtained by always choosing the first successor (conditions (1) of Definition 7 and (3) of Definition 8). Moreover, $\mathcal{A}_E$ checks conditions (2) of Definition 8. It is defined such that:

- S_E is the set of triples $(\mathcal{H}, E, t)$ where $\mathcal{H}$ and E play the same role as in $\mathcal{A}_J$ and t is a Boolean value ($\top$ or $\bot$) indicating whether the state is final;
- $S_{E,0} = \{(\emptyset, \emptyset, \bot)\}$ and $F_E = \{(S_E, F)\}$ where $F = \{(\mathcal{H}, E, t) \in S_E \mid t = \top\}$.

The transition function ρ_E is defined such that if

$$((\mathcal{H}_1, E_1, t_1), \ldots, (\mathcal{H}_{N^{\equiv}_{\varphi_0}}, E_{N^{\equiv}_{\varphi_0}}, t_{N^{\equiv}_{\varphi_0}})) \in \rho_E((\mathcal{H}_0, E_0, t_0), (P, \mathcal{C}))$$

then all the following conditions hold:

- for all $k \in 1..N^{\equiv}_{\varphi_0}$, either $\mathcal{H}_k \in \mathcal{C}$ or $\mathcal{H}_k = FL(\varphi_0)$;
- if $\mathcal{H}_0 = \emptyset$ then $\mathcal{H}_1$ is a Hintikka set, $\varphi_0 \in \mathcal{H}_1$ and $P = \emptyset$;
- if $\mathcal{H}_0$ is a Hintikka set then $P \neq \emptyset$, $\mathcal{H}_1$ is a Hintikka set and $\mathcal{H}_0\ S_P\ \mathcal{H}_1$;
- if $\mathcal{H}_1$ is a Hintikka set and $E_0 = \emptyset$ then for all eventuality $\langle\!\langle\alpha\rangle\!\rangle\varphi \in \mathcal{H}_1$ there is $(G_2, \eta_2) \in \text{next}(\alpha\varphi)$ such that $G_2 \cup \{\text{form}(\eta_2)\} \subseteq \mathcal{H}_1$, if $\eta_2^1 \in Atm$ then $\eta_2^{\geq 2} \in E_1$ and if $\eta_2^1 = \equiv$ and $E_k \neq \emptyset$ for k such that $\text{form}(\eta_2) = \langle\!\langle\equiv\rangle\!\rangle\psi_{k-1}$ then $t_k = \bot$;
- if $\mathcal{H}_1$ is a Hintikka set then for all $\eta_1 \in E_0$, $\text{form}(\eta_1) \in \mathcal{H}_1$ and there is $(G_2, \eta_2) \in \text{next}(\eta_1)$ such that $G_2 \cup \{\text{form}(\eta_2)\} \subseteq \mathcal{H}_1$, if $\eta_2^1 \in Atm$ then $\eta_2^{\geq 2} \in E_1$ and if $\eta_2^1 = \equiv$ and $E_k \neq \emptyset$ for k such that $\text{form}(\eta_2) = \langle\!\langle\equiv\rangle\!\rangle\psi_{k-1}$ then $t_k = \bot$;
- for all $k \in 2..N^{\equiv}_{\varphi_0}$, if $\mathcal{H}_1$ is a Hintikka set and $\langle\!\langle\equiv\rangle\!\rangle\psi_{k-1} \in \mathcal{H}_1$ then $\mathcal{H}_k$ is a Hintikka set, $\psi_{k-1} \in \mathcal{H}_k$ and there is $(G_2, \eta_2) \in \text{next}(\eta_1)$ where η_1 is the maximal eventuality chain for ψ_{k-1} such that $G_2 \cup \{\text{form}(\eta_2)\} \subseteq \mathcal{H}_k$ and if $\eta_2^1 \in Atm$ then $\eta_2^{\geq 2} \in E_k$;
- if $E_1 \neq \emptyset$ then $t_1 = \bot$.

$\mathcal{A}_S$ is deterministic and the number of its states is double exponential in $|\varphi_0|$. It can be directly translated into a Streett tree automaton with no termination pair. $\mathcal{A}_J$ has an exponential number of states but it must be determinized before being transformed into a tree automaton, because the choice of the Hintikka sets depends on the successor of the node. By the construction of Piterman [11], any nondeterministic Büchi word automaton with s states can be transformed into an equivalent deterministic Streett word automaton with s^{2s+2} states and s pairs. Hence, the resulting Streett tree automaton corresponding to $\mathcal{A}_J$ has a double exponential number of states and an exponential number of termination pairs. $\mathcal{A}_E$ has an exponential number of states and a single termination pair. The product of these three tree automata gives a Streett tree automaton $\mathcal{A}$ with a double exponential number of states and an exponential number of pairs. Emerson and Jutla [8] proved that the emptiness of a Streett tree automaton with s states and p termination pairs can be decided in deterministic time $(s \cdot p)^{\mathcal{O}(p)}$.

Since $\mathcal{A}$ recognizes exactly the good syntactic trees for φ_0, by Theorem 4, the satisfiability problem of OPDL^{lc} is in 2EXPTIME. Moreover, the proof from [2] that OPDL^{lts} is 2EXPTIME-hard can easily be adapted to OPDL^{lc}. Hence we have the following result.

Theorem 5. *The satisfiability problem of OPDL^{lc} is 2EXPTIME-complete.*

5 Conclusion

In this work, we have first shown that the logic OPDL^{lts} proposed by [2] does not have the good property of being conservative. Using the more convenient path semantics framework, the semantics of this logic has been slightly modified to obtain the new logic OPDL^{lc} which is conservative and in which PDL and CTL^* can still be embedded. Then, we have answered the question, left open in [2], of the complexity of the satisfiability problems of OPDL and OPDL^{lc}. We have proved that both problems are 2EXPTIME-complete. However, the methods used to prove these results are quite different. Whereas for OPDL a finite model with bounded size is constructed, for OPDL^{lc} infinite branches must be considered using automata on infinite trees. This highlights the difference between OPDL and OPDL^{lc} as a consequence of the limit closure property of the path semantics.

Some questions about OPDL and OPDL^{lc} have been left open for future research. For instance, there is still no axiomatization for OPDL and OPDL^{lc}. Furthermore, it would be interesting to study the relative expressive power of these logics and other logics embedding both PDL and CTL^* like the automata-based logic YAPL [18] or the extension PDL$-\Delta$ of PDL with repetition [16].

Another issue of future research is the relation between OPDL, OPDL^{lc} and ATL^*, the full version of Alternating-time Temporal Logic (ATL) introduced in [1]. There have recently been interesting results by [6], providing a tableau-based decision procedure for ATL^*, which has been proved to be in 2EXPTIME as well, and to also work for CTL^*. The procedure has been implemented. Future research will be devoted to verify whether a similar solution can be found for OPDL and OPDL^{lc} in order to have an implemented procedure for checking satisfiability in these logics.

References

1. Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002)
2. Balbiani, P., Lorini, E.: Ockhamist propositional dynamic logic: a natural link between PDL and CTL*. In: Libkin, L., Kohlenbach, U., Queiroz, R. (eds.) WoLLIC 2013. LNCS, vol. 8071, pp. 251–265. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39992-3_22
3. Belnap, N., Perloff, M., Xu, M.: Facing the Future: Agents and Choices in Our Indeterminist World. Oxford University Press, New York (2001)

4. Brown, M., Goranko, V.: An extended branching-time Ockhamist temporal logic. J. Logic Lang. Inform. **8**(2), 143–166 (1999)
5. Dam, M.: CTL* and ECTL* as fragments of the modal mu-calculus. Theoret. Comput. Sci. **126**(1), 77–96 (1994)
6. David, A., Schewe, S.: Deciding ATL* satisfiability by tableaux. Technical report, Laboratoire IBISC - Université d'Evry Val-d'Essonne (2016)
7. Emerson, E., Sistla, A.: Deciding full branching time logic. Inf. Control **61**, 175–201 (1984)
8. Emerson, E.A., Jutla, C.S.: The complexity of tree automata and logics of programs. SIAM J. Comput. **29**(1), 132–158 (1999)
9. Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. J. Comput. Syst. Sci. **18**(2), 194–211 (1979)
10. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
11. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. In: Logic in Computer Science (LICS), pp. 255–264. IEEE Computer Society (2006)
12. Pratt, V.R.: Models of program logics. In: 20th Annual Symposium on Foundations of Computer Science, pp. 115–122. IEEE Computer Society (1979)
13. Prior, A.: Past, Present, and Future. Clarendon Press, Oxford (1967)
14. Reynolds, M.: An axiomatization of full computation tree logic. J. Symbol. Logic **66**(3), 1011–1057 (2001)
15. Reynolds, M.: A tableau for bundled CTL*. J. Logic Comput. **17**(1), 117–132 (2007)
16. Streett, R.S.: Propositional dynamic logic of looping and converse is elementarily decidable. Inf. Control **54**(1–2), 121–141 (1982)
17. Thomason, R.: Combinations of tense and modality. In: Gabbay, D., Guenthner, F. (eds.) Handbook of Philosophical Logic, vol. 2, 2nd edn, pp. 135–165. Reidel, Dordrecht (1984)
18. Vardi, M.Y., Wolper, P.: Yet another process logic (preliminary version). In: Clarke, E., Kozen, D. (eds.) Logic of Programs 1983. LNCS, vol. 164, pp. 501–512. Springer, Heidelberg (1984). doi:10.1007/3-540-12896-4_383
19. Wolper, P.: A translation from full branching time temporal logic to one letter propositional dynamic logic with looping (unpublished manuscript)
20. Zanardo, A.: Branching-time logic with quantification over branches: the point of view of modal logic. J. Symbol. Logic **61**(1), 143–166 (1996)