Chain-Monadic Second Order Logic over Regular Automatic Trees and Epistemic Planning Synthesis

Gaëtan Douéneau-Tabot
ENS Paris-Saclay, Université Paris-Saclay

Sophie Pinchinat
IRISA, Université de Rennes

François Schwarzentruber
IRISA, ENS Rennes

Abstract

We consider infinite relational structures that have a finite presentation by means of a finite tuple of finite-state automata, and already known as automatic structures. While it is well established that model checking against first-order logic is decidable over automatic structures, we show how this seminal result can be adapted for a restricted class of automatic structures called regular automatic trees and an extension of the logic based on chain-MSO, and written cMSO (where MSO stands for monadic second-order logic). The logic cMSO, as chain-MSO, is interpreted over trees, and its second-order quantifiers range over subsets of branches. In the setting of regular automatic trees, we relate cMSO and logics of knowledge and time, among which the branching epistemic linear-time mu-calculus BLlinK. We finally apply our results to dynamic epistemic logic and its related epistemic planning problem: when restricting to event models with propositional preconditions and postconditions, the relational structures arising from epistemic planning problems turn out to be regular automatic trees. This already established latter central property allows us to derive the (already known) decidability of the epistemic planning problem as a mere corollary, but also to enlarge the class of decidable epistemic planning problems to goals expressed in BLlinK, with an effective way of computing the set of all successful (possibly infinite) plans.

Keywords: Automatic structures, chain Monadic Second-Order logic, Epistemic planning synthesis.

1 gaetan.doueneau@ens-paris-saclay.fr
2 sophie.pinchinat@irisa.fr
3 francois.schwarzentruber@ens-rennes.fr
1 Introduction

In artificial intelligence, classical planning consists in generating a finite sequence of actions for achieving a given goal. The so-called epistemic planning [6] generalizes classical planning with epistemic goal (agent $a$ knows that...) and complex actions (public announcements, private announcements, etc.). Epistemic planning is based on Dynamic epistemic logic ([2], [25]) and is undecidable in the general case [7].

Actions are represented by event models 4 that are Kripke structures of events, instead of possible worlds, equipped with a precondition and a postcondition. When event preconditions and postconditions are propositional (Boolean) only (intuitively, announcements are Boolean formulas and uncertainty is only about the content of the messages and physical changes), epistemic planning has been shown to be decidable ([30],[15]).

Regrettably, epistemic planning as considered so far concerns finite plans, and is in essence a reachability problem. However, real-life applications may challenge infinite plans. We give three examples of planning goals that require an infinite plan.

(i) safety properties, e.g. 'invariantly, an intruder $a$ does not know the location of the piece of jewelry more than 3 consecutive steps'.

(ii) recurring bounded properties, e.g. 'all drones know that the region is safe every 20 steps';

(iii) strategic reasoning, e.g. 'with the current plan, the drone $a$ never knows the region is safe but every 10 steps, there is a(nother) plan to let the drone $a$ eventually know the region is safe'.

Nevertheless, infinite plans have been considered in [15] in the DEL (Dynamic Epistemic Logic) setting. In this approach, DEL structures, those arising from iterating ad infinitum the triggering of events, are seen as infinite relational structures. Maubert has shown that, when all event preconditions and postconditions are propositional, DEL structures are automatic structures [5,18]. Automatic structures are relational structures which have a presentation by means of a finite family of finite-state automata. As noticed by Maubert, the decidability result of epistemic planning (for epistemic goals) relies on the fact that model checking against first-order logic is decidable over automatic structures: in a DEL structure, the skeleton is a tree whose nodes (or equivalently finite branches) are finite plans, i.e. sequences of triggered events, and this tree structure is augmented with “transverse” binary relations between nodes in possibly different branches. These extra relations denote epistemic relations between histories. In DEL structures, the existence of a plan reduces to the existence of a point in this automatic (tree) structure where the epistemic planning goal (hence expressible in FO) holds. After all, the whole epistemic planning problem can be expressed in FO whose outermost quantifier is ex-

4 also called sometimes “action models” in the literature.
istential. Additionally, automata constructions that answer the query of an FO-formula on an automatic structure allow to synthesize an automaton that recognizes the set of all (finite) plans achieving the goal. Regarding infinite plans, Maubert relies on the involved theory of uniform strategies he has developed [15]. This theory enables him to deal with epistemic planning instances whose goals are expressed in the branching-time epistemic logic $\text{CTL}^*K$, the extension of temporal logic $\text{CTL}^*$ [12] with epistemic modalities. The setting does lead to synthesizing infinite plans, but the synthesis relies on a bottom-up technique over the goal formula that prevents the setting from being extended to logics featuring fix-points; while statements (ii) and (iii) above cannot be expressed in $\text{CTL}^*K$, a well-tuned logic with fix-point or monadic with second-order quantifiers would suffice.

In an attempt to enlarge the specification language for epistemic planning goals, that goes beyond $\text{CTL}^*K$ while remaining decidable, one should observe that using the full monadic second-order logic (MSO) is hopeless: by [22], model checking against MSO over the binary tree equipped with the transverse binary relation “equal level” is undecidable. Relation “equal level” is an instance of an agent epistemic relation where the agent does not observe anything from the current finite sequence of triggered events but its length. However, there are variants of MSO, where second-order quantifications are constrained enough to yield a decidable model checking problem over infinite trees of the kind of DEL structures. For example, relying on [12], and as depicted in Figure 1, there exist mainly three interpretations of the second-order quantifiers, yielding full MSO, path-MSO, and chain-MSO. The full MSO logic can be interpreted over arbitrary relational structures and the second-order quantifiers range over all subsets of the structure domain. On the contrary, the path-MSO logic is interpreted on (possibly infinite) trees and the quantifiers range over paths/branches\(^5\) of the tree. Finally, chain-MSO, also interpreted on trees only, requires the second-order quantifier to range over chains, that are subsets of paths [12]. It is known that the expressive power of chain-MSO strictly subsumes the one of path-MSO, the former allowing to describe arbitrary $\omega$-regular properties of branches, while the latter captures only star-free properties, see [12].

In this paper, we extend chain-MSO of [21,12] with relations in the tree, e.g. interpreting epistemic modalities, making statements (ii) and (iii) expressible. This extension is written cMSO. We introduce regular automatic trees, a strict but large subclass of automatic structures that encompass DEL structures with propositional events. We then show that model checking against cMSO is decidable over regular automatic trees. Our decidability result is strongly inspired from the proof technique in [23, Th. 5.2] to show that chain-MSO with the “equal level” predicate is decidable over $n$-ary trees. Our proof yields automata constructions that can be exploited to achieve the synthesis of plans in epistemic planning.

\(^5\) The difference between paths and branches is irrelevant by [12].
4 Chain-MSO Logic over Regular Automatic Trees and Epistemic Planning Synthesis

![Figure 1](image)

(a) MSO quantification over any subset.
(b) path-MSO quantification over any path in a tree.
(c) cMSO quantification over any chain in a tree.

Figure 1. Different restrictions of second-order quantifications.

Noticeably, the logic cMSO captures the linear-time mu-calculus ([4], [26]) enriched with path quantifiers and epistemic modalities, here written BL$_{\text{lin}}^\mu$K (the acronym stands for ‘Branching Logic of the linear-time $\mu$-calculus with Knowledge). In essence, the logic BL$_{\text{lin}}^\mu$K is an epistemic extension of the logic ECTL$^*$, for “extended CTL$^*$”, of [12], but in a mu-calculus style rather than with the use of automata modalities, in the sense of [29]. Since CTL$^*$K can easily be embedded into BL$_{\text{lin}}^\mu$K (the hard-coded linear-time temporal operators of CTL$^*$ are based on fix-points expressible in the linear-time mu-calculus), our result significantly exceeds the one of [15] whose proof technique does not allow for considering arbitrary fix-point formulas.

The presented contribution is derived from the preliminary work available in [10]. In Section 2, we recall the notion of automatic structures and results on model checking against FO-formulas and MSO-formulas. In Section 3, we present the logic cMSO, as well as the subclass of so-called regular automatic trees. We prove that model checking against cMSO is decidable over this subclass (Theorem 3.9). In Section 4, we compare cMSO with some logics of knowledge and time, namely CTL$^*$K, CTL$^*$ and BL$_{\text{lin}}^\mu$K, with increasing expressive power. Section 5 is dedicated to the application of our results to the generalized epistemic planning problem (Definition 5.7) where goals are arbitrary BL$_{\text{lin}}^\mu$K-statements and instances of the planning domain rely on DEL specifications with propositional events.

2 Automatic structures and decidable theories

In this section we recall seminal results on automatic structures: namely that first-order logic FO is decidable over these structures while monadic second-order logic MSO is not. The interested reader may refer to [5,18] for further details.

2.1 Structures and logics

Logics FO and MSO are interpreted over relation structures.

**Definition 2.1** A relational structure is a structure of the form $A = \langle D, R_1 \ldots R_p \rangle$ where $D$ is a non-empty set called the domain; $R_1 \ldots R_p$ are relations over $D$ of arity $r_1 \ldots r_p$, respectively; namely $R_i \subseteq D^{r_i}$.
The set of symbols \( \{ R_1, \ldots, R_p \} \) is called the signature of \( A \). We take the convention to write \( R_i(d_1, \ldots, d_r) \) for \( (d_1, \ldots, d_r) \in R_i \). We distinguish the particular class of structures that are infinite trees of fixed finite degree \( n \), called \( n \)-ary trees, that are central in our contribution. These structures represent computation trees (with a root of address \( \varepsilon \), \( i \)-successor relations, and prefix binary relation between addresses) augmented with additional relations, such as epistemic relations.

**Definition 2.2** Given a finite set \( E := \{ 1, \ldots, n \} \) of directions, we call \( n \)-ary tree a structure \( T = \langle D, r, S_1, \ldots, S_n, R_1, \ldots, R_p \rangle \) where:

- \( D \) is a prefix-closed subset of \( E^* \) (the addresses of the nodes);
- \( r \) is a unary relation that holds only for \( \varepsilon \), the root address of the tree;
- \( S_e(x, xe) \) whenever \( xe \in D \);
- \( R_1, \ldots, R_p \) are additional relations on \( D \).

In the particular case of a tree, elements of \( D \) are nodes.

Notice that in Definition 2.2 of a tree no condition was imposed on the relations \( R_1 \ldots R_p \). They may therefore correspond to transversal links between nodes just as epistemic relations would do, e.g. the “equal level” binary relation.

**Example 2.3** The structure \( T_2 = \langle E^*, S_1, S_2 \rangle \) is a 2-ary tree called the infinite full binary tree. Also the structure \( T_2^1 = \langle E^*, S_1, S_2, el \rangle \) obtained by augmenting \( T_2 \) with the binary relation “at equal level in the tree” (\( el(x, y) \) holds if, and only if, \( |x| = |y| \)) is another example of a 2-ary tree.

First-order logic (FO) over relational structures \( A = \langle D, R_1 \ldots R_p \rangle \) concerns formulas that conform to the following syntax: \( \varphi := R_i(x_1 \ldots x_r) \mid \neg \varphi \mid (\varphi \land \varphi) \mid \exists x \varphi \), where \( x, x_1, x_r \) are first-order variables whose interpretation ranges over the domain of relational structures.

When turning to the more expressive monadic second-order logic (MSO), one allows for second-order variables that range over subsets of the domain of relational structures. Formally, the syntax of MSO is given by: \( \varphi := R_i(x_1 \ldots x_r) \mid x \in X \mid \neg \varphi \mid (\varphi \land \varphi) \mid \exists x \varphi \mid \exists X \varphi \), where \( x \) is a first-order variable and \( X \) is a second-order variable whose interpretation ranges over subsets of the domain. In the following, we let \( V_1 \) and \( V_2 \) denote respectively the set of first-order variables and second-order variables.

As usual, in a formula, a variable \( x \in V_1 \) (resp. \( X \in V_2 \)) is free if it is not under the scope of a quantifier, namely some \( Qx \) (resp. \( QX \)) with \( Q \in \{ \forall, \exists \} \). A formula is closed if it contains no free variables. An assignment in domain \( D \) is a function \( \sigma \) that maps a variable of \( V_1 \) onto an element of \( D \) and a variable of \( V_2 \) onto a subset of \( D \). Given an assignment \( \sigma \) in \( D \), a variable \( x \in V_1 \) and \( d \in D \), we let \( \sigma[x \mapsto d] \) be the assignment on \( D \) that coincides with \( \sigma \) but maps \( x \) onto \( d \). Similarly, we define \( \sigma[X \mapsto D'] \) where \( D' \subseteq D \) for the second-order variables.

Due to limited space, we do not provide the semantics of MSO, see for example [11].
2.2 Automatic presentations

We now describe how some relational structures can be encoded using formal languages, following the presentation of [18]. By alphabet we mean a finite set of symbols, named letters. A word over $\Sigma$ is a finite sequence of letters. We denote by $\Sigma^*$ the set of such sequences. For $u \in \Sigma^*$ and $n \geq 0$, let $u[n]$ be the $n+1$-th letter of $u$ (when defined). We assume familiarity with the basic definitions of automata theory and the properties of regular languages.

Let $\square$ be a fresh padding symbol. If $\Sigma$ is an alphabet, we define $\Sigma\square$ as $\Sigma \cup \{\square\}$. The following definition describes an encoding for $k$-tuples of words as a word on the product alphabet $\Sigma_0^k$; since words of the tuple may have different length, the padding symbol $\square$ is used to align the words.

**Definition 2.4** If $u, v \in \Sigma^*$, their convolution $u \otimes v$ is the word of $(\Sigma\square \times \Sigma\square)^*$ of length $\max(|u|, |v|)$ such that $(u \otimes v)[n] = (u[n], v[n])$ if $n < \min(|u|, |v|)$; $(u \otimes v)[n] = (u[n], \square)$ if $|v| \leq n < |u|$; and $(u \otimes v)[n] = (\square, v[n])$ if $|u| \leq n < |v|$. Convolution is defined similarly for $k$-tuples of words.

**Example 2.5** The convolution of $aabab \otimes b \otimes ba$ over alphabet $\Sigma = \{a, b\}$ is the four-letter word $abab \otimes b \otimes ba$ over alphabet $\Sigma_0^3$.

**Definition 2.6** Let $A = \langle D, R_1 \ldots R_p \rangle$ be a relational structure. An automatic presentation of $A$ is a tuple of regular languages $(L_D, L_1, \ldots, L_n)$ meeting the following conditions: (1) there exists a one-to-one encoding function $\text{enc} : D \rightarrow L_D$, and (2) for all $R_i$ (arity $r_i$), $L_i = \{u_1 \otimes \cdots \otimes u_{r_i} \mid \forall j, u_j \in L_D \text{ and } (\text{enc}^{-1}(u_1), \ldots, \text{enc}^{-1}(u_{r_i})) \in R_i\}$.

The alphabet $\Sigma$ of $L_D$ is called the encoding alphabet. The inverse $\text{enc}^{-1}$ of the encoding function is the decoding function. We write $\text{enc}(D)$ for $L_D$, and more generally, $\text{enc}(R)$ for the set $\{\text{enc}(d_1) \otimes \cdots \otimes \text{enc}(d_e) \mid (d_1, \ldots, d_e) \in R\}$, for an arbitrary relation $R \subseteq D^e$.

A structure is automatic if it has (at least) an automatic presentation. A presentation can effectively be described by a tuple of finite automata $(M_D, M_1, \ldots, M_p)$ recognizing $(L_D, L_1, \ldots, L_n)$.

**Remark 2.7** We may assume that equality is among the relations $R_i$, represented by the regular language $\{u \otimes u \mid u \in L_D\}$. In the literature, the standard definition of automatic presentations allows an element to have several encodings, whenever equality can be presented by some regular language. However, both definitions enable to present the same structures [5].

**Example 2.8** (i) Finite structures are automatic, since finite languages are regular.

(ii) $(\mathbb{N}, \leq)$ is automatic, with an automatic presentation over the unary alphabet $\{1\}$ by letting $\text{enc}(n) = 1^n$. Automaton $M_{\leq}$ of Figure 2 verifies there are less 1’s in the first component than in the second component.

(iii) Both trees $T_2$ and $T_2^R$ are automatic.
We recall the known fundamental theorem of [18, Th. 3.1] regarding the model checking problem against \( \text{FO} \) over automatic structures.

**Definition 2.9** Model checking against \( \text{FO} \)

**Input:** A presentation \((M_D, M_1, \ldots, M_p)\) of a relational structure \(A\) and a closed \(\text{FO}\)-formula \(\varphi\) over the corresponding signature.

**Output:** Yes if \(A \models \varphi\), no otherwise.

**Theorem 2.10** [18, Th. 3.1] Model checking against \( \text{FO} \) is decidable.

The main ingredient of the proof of Theorem 2.10 relies on the construction of Proposition 2.11 which heavily relies on the closure of regular languages by intersection, negation and projection over components.

Let \(\varphi(x_1 \ldots x_n) \in \text{FO}\) where \(x_1 \ldots x_n \in V_1\) are the only free variables, and let \(A\) be a relational structure. We write \(\varphi^A := \{(d_1 \ldots d_n) \in D^n \mid A[x_i \mapsto d_i]_1 \leq i \leq n \models \varphi[x_1 \ldots x_n]\}\) for the set of tuples that satisfy \(\varphi\).

**Proposition 2.11** There is an algorithm that given an automatic presentation of a relational structure \(A\) with encoding function \(\text{enc}\), and a first-order formula \(\varphi(x_1, \ldots, x_n)\), outputs an automaton that recognizes \(\text{enc}(\varphi^A)\).

Theorem 2.10 can sometimes be extended to \(\text{MSO}\). For example, there are very specific cases where \(\text{MSO}\) can be decided, among which the typical example of the automatic structure \((\mathbb{N}, <)\).

**Proposition 2.12** [3] The \(\text{MSO}\)-theory of a structure having an automatic presentation based on a unary encoding alphabet is decidable.

Also, by Rabin’s Theorem [17], \(\text{MSO}\) is decidable over the full binary tree with signature \(r, S_1, S_2\). However, \(\text{MSO}\) becomes undecidable over the full binary relation augmented with the “equal level” relation \(el\) (see Example 2.3), although the obtained structure is automatic.

**Theorem 2.13** [22] Model checking against \(\text{MSO}\) over the binary tree \(T_2^{el}\) is undecidable.

We now focus on a subclass of automatic structures where the undecidability frontier can be pushed back.

### 3 Model checking against \(c\text{MSO}\) over RA trees

We consider the variant of \(\text{MSO}\) stemming from “chain \(\text{MSO}\)”, written \(c\text{MSO}\) in this paper, which contrary to \(\text{FO}\) and \(\text{MSO}\), can only be interpreted over
infinite trees. The second-order quantifiers range over chains, namely subsets of nodes along a branch, as depicted in Figure 1(c). The logic \( \text{cMSO} \) has already been defined in [21,12,23] for \( n \)-ary trees and with a signature restricted to the successor relations \( S_1, \ldots, S_n \); the main results concern the decidability of the model checking of \( \text{chain-MSO} \) over the full binary tree, and its expressivity. In our setting, the signature of \( \text{cMSO} \) can be arbitrary, so that properties involving e.g. knowledge and time can be expressed.

We also exhibit a subclass of automatic structures called regular automatic trees that are automatic trees where the encoding of a node is its address in the tree. This subclass is large and encompasses infinite models arising from the unfolding of finite-state systems but also in dynamic epistemic logic. We establish that model checking against \( \text{cMSO} \) is decidable over regular automatic trees (Theorem 3.9).

3.1 The logic \( \text{cMSO} \) over trees

We recall a tree structure \( T = \langle D, r, S_1, \ldots, S_n, R_1, \ldots, R_p \rangle \) over a finite set \( E = \{1, \ldots, n\} \) of directions (Definition 2.2), where \( D \) is a prefix-closed subset of \( E^* \) describing the nodes through their addresses in the tree. In the following, we let \( S^* \) be the reflexive and transitive closure of the generalized successor relation \( S := \bigcup_{i=1}^n S_i \).

**Definition 3.1** In a tree structure \( T \) of domain \( D \subseteq E^* \), a subset \( C \subseteq D \) is a chain if it is totally ordered with respect to \( S^* \), i.e. for all \( d, d' \in C \), either \( S^*(d, d') \) or \( S^*(d', d) \) holds.

We denote by \( \text{Ch}(T) \) the set of chains of the tree \( T \).

**Example 3.2** In the full binary tree \( T_2 \), the set \( \{1^n \mid n \in \mathbb{N}\} \) is a chain, whereas \( \{1, 2\} \) is not.

The notion of chains enables one to consider an interpretation of MSO formulas in trees, named cMSO, where the second-order quantifiers are restricted to chains.

**Definition 3.3** The logic \( \text{cMSO} \) is interpreted in a tree \( T = \langle D, r, S_1, \ldots, S_n, R_1, \ldots, R_p \rangle \) with an assignment \( \sigma \) in \( D \) for free variables as follows.

\[
\begin{align*}
T, \sigma &\models R_i(x_1 \ldots x_{r_i}) \iff (\sigma(x_1) \ldots \sigma(x_{r_i})) \in R_i; \\
T, \sigma &\models x \in X \quad \iff \sigma(x) \in \sigma(X); \\
T, \sigma &\models \neg \varphi \quad \iff T, \sigma \not\models \varphi; \\
T, \sigma &\models \varphi \land \psi \quad \iff T, \sigma \models \varphi \text{ and } T, \sigma \models \psi; \\
T, \sigma &\models \exists x \varphi \quad \iff \text{there exists } d \in D \text{ such that } T, \sigma[x \mapsto d] \models \varphi; \\
T, \sigma &\models \exists X \varphi \quad \iff \text{there exists } C \in \text{Ch}(T) \text{ s. th. } T, \sigma[x \mapsto C] \models \varphi.
\end{align*}
\]

For a closed formula \( \varphi \), we simply write \( T \models \varphi \) without specifying the assignment.
The logic \(c\text{MSO}\) is close to the logic \(\text{path-MSO}\) \([21,12]\) which restricts second-order quantification to maximal branches only. Actually the latter is a subsystem of the former: the \(c\text{MSO}\)-formula \(\text{path}\{X, x_0\} := x_0 \in X \land \forall x \{ x \in X \to [(\exists y S(x, y) \to \exists y (S(x, y) \land y \in X)) \land \neg S(x, x_0)]\} \) expresses that chain \(X\) is a maximal path starting at note \(x_0\), which corresponds to the very quantifier in \(\text{path-MSO}\). This translation has already been noticed many times in the literature.

### 3.2 Regular automatic trees

We now turn to a particular class of trees, called regular automatic trees, which are automatic trees for the most intuitive encoding of nodes, that is by their address in the tree. Since the domain is required to be regular, such trees arise from the unfolding of some finite-state transition systems, hence the terminology of regular trees.

**Definition 3.4** A tree \(\mathcal{T} = (D, r, S_1, \ldots, S_n, R_1, \ldots, R_p)\) on the set of directions \(E = \{1, \ldots, n\}\) is a regular automatic (RA) tree if the identity function \(\text{enc} : D \to E^*\) describes an automatic presentation of \(\mathcal{T}\) over the encoding alphabet \(E\). This presentation is called the canonical presentation of \(\mathcal{T}\).

Since the encoding function of a RA tree is the identity function, the regularity of \(S_1, \ldots, S_n\) and \(\preceq\) directly follows from that of \(D\). Thus, a tree is a regular automatic tree if, and only if, \(D \subseteq E^*\) is regular, and for all \(R_i\) (of arity \(r_i\)), \(L_i = \{d_1 \otimes \cdots \otimes d_{r_i} \mid (d_1, \ldots, d_{r_i}) \in R_i\}\) is regular.

**Example 3.5** The tree \(\mathcal{T}^{el}_2\) of Example 2.3 is a regular automatic tree.

As stated in the following proposition, RA trees form a strict subclass of automatic trees.

**Proposition 3.6** There are automatic trees that are not RA trees.

**Proof**

Let \(E = \{1, 2\}\) we consider the binary tree \(\mathcal{T}^{auto}_{notreg} = (D, r, S_1, S_2, el)\) of Figure 3(a) where \(D = \{1^m2^k \mid 0 \leq k \leq m\}\) and \(el\) is the “equal level” relation of Example 2.3.

![Figure 3. An automatic tree that is not regular.](image)

As \(D\) is not a regular language, \(\mathcal{T}^{auto}_{notreg}\) is not a RA tree, but it is automatic: consider the encoding alphabet \(\Sigma = (\{0, 1\} \square)^2\) and let \(\text{enc}(1^m2^k) = \text{bin}(m) \otimes\)
bin(k) be the encoding function, as presented in Figure 3(b), where bin(m) (resp. bin(k)) the binary representation of m (resp. k) with least significant digit first. For example, \(\text{enc}(112) = \text{enc}(1^22^1) = \text{bin}(2) \otimes \text{bin}(1) = (\binom{1}{2})\binom{1}{2}.\) The reader should easily get convinced that enc(D), enc(S_1), enc(S_2) and enc(el) are regular languages. □

When restricting to RA trees, some usual relations over trees can be easily captured by automata over the canonical encoding of nodes. We recall the following additional binary relations over trees.

- The reflexive and transitive closure \(S^*\) of the generalized successor binary relation \(S;\)
- The binary relation \((d,d') \in \lessdot\) whenever node \(d\) is not deeper than node \(d'\) in \(T;\)
- The binary relation el, where \((d,d') \in \text{el}\) whenever nodes \(d\) and \(d'\) are at the same depth in the tree.
- The binary equality relation =.

**Lemma 3.7** Let \(\langle D, r, S_1, \ldots, S_n, R_1, \ldots, R_p \rangle\) be a RA tree. Then the relational structure \(\langle D, r, S_1, \ldots, S_n, R_1 \ldots R_p, S^*, \lessdot, \text{el}, = \rangle\) is also an RA tree.

Restricting to the class of RA trees allows for decidability results of model checking against logics that go beyond FO.

### 3.3 Model checking against cMSO

We now describe the main result of this section, as stated by Theorem 3.9, where the model checking problem against cMSO is as follows.

**Definition 3.8** Model checking against cMSO

**Input**: \(\langle M_D, M_r, (M'_i)_{1 \leq i \leq n}, (M_i)_{1 \leq i \leq p} \rangle\) the canonical presentation of a RA tree \(T\) and a closed cMSO-formula \(\varphi\) over the corresponding signature.

**Output**: Yes if \(T \models \varphi\), no otherwise.

Theorem 3.9 can be read as a variant of Theorem 2.10 where, on the one hand, the logic FO is extended, and, on the other hand, the class of automatic structures is restricted. It can also be read as a generalization of [23, Th. 5.2].

**Theorem 3.9** Model checking against cMSO is decidable over RA trees.

The rest of this section is dedicated to the proof of Theorem 3.9. This proof goes over the proof made in [23, Th. 5.2] to generalize the result to arbitrary RA trees.

Before starting the proof, we may assume without loss of generality, that all the variables occurring in the formulas are second-order variables. This means to add an extra unary predicate Sing() which conveys that a set is a singleton, and to add inclusion formulas of the form \(X \subseteq Y\) so as to capture previous formulas of the form \(x \in X\). The reader may refer to [23] for this routine
transformation. The syntax then becomes as follows.

\[ \varphi := \text{Sing}(X) \mid X \subseteq Y \mid R((X_1 \ldots X_n)) \mid \lnot \varphi \mid (\varphi \land \varphi) \mid \exists X \varphi, \]
\[(1) \]

where \( X, Y, X_1, \ldots \in V_2 \).

The new syntax requires to adjust the semantics of the formulas, and in particular of those of the form \( R((X_1 \ldots X_n)) \), by imposing that each \( X_j \) is interpreted as a singleton, this is routine. The logic resulting from (1) is expressively equivalent to cMSO to the extent that a singleton is a chain. Therefore, in the following, we still use cMSO when using the syntax of (1).

A key idea to develop automata construction to model check against cMSO is that a set of addresses is a chain if, and only if, it is contained in the set of all prefixes of some infinite word. Given a chain \( C \subseteq E^* \), we let \( \text{Branches}(C) := \bigcap \{uE^\omega \mid u \in C \} \) be the set of infinite words whose set of prefixes contains \( C \). In particular, \( \text{Branches}(C) \) is a singleton if, and only if, \( C \) is infinite.

For \( v \in \text{Branches}(C) \), we write \( m\text{branch}(v, C) \) for the infinite word over \( E \times \{0, 1\} \) whose first component is \( v \) and whose second component is marked by 1 if the current prefix of \( v \) belongs to \( C \), and by 0 otherwise. Formally, \( m\text{branch}(v, C) = m[0]m[1]m[2] \ldots \) with \( m[i] = (v[i], b) \) where \( b = 1 \) if \( v[0]v[1] \ldots v[i] \in C \), and \( b = 0 \) otherwise.

The words \( m\text{branch}(v, C) \) are used to encode the chain \( C \). We will overload the encoding identity function \( \text{enc} \) of RA trees to denote the encoding of chains. We let \( \text{enc}(C) := \{m\text{branch}(v, C) \mid v \in \text{Branches}(C)\} \). Also, we extend the notation to \( \text{enc}(C_1, \ldots, C_m) \) that contains infinite words over alphabet \( (E \times \{0, 1\})^m \) by letting \( \text{enc}(C_1, \ldots, C_m) := \text{enc}(C_1) \otimes \ldots \otimes \text{enc}(C_m) \) where \( \otimes \) is the convolution extended to sets of infinite words in a natural manner. For \( m = 0 \), \( \text{enc}() \) is the singleton containing the unique infinite word over the singleton alphabet \( (\Sigma \times \{0, 1\})^0 \).

We now provide an automata-theoretic approach for model checking against cMSO formulas. We first establish the existence of a Büchi automaton \( C_m \) that verifies if an infinite word over alphabet \( (E \times \{0, 1\})^m \) denotes a \( m \)-tuple of chains, as stated by Lemma 3.10.

**Lemma 3.10** One can effectively construct a Büchi automaton \( C_m \) that recognizes the encoding of \( m \)-tuples of chains, namely the language

\[ \bigcup_{C_1, \ldots, C_m \in \text{Ch}(T)} \text{enc}(C_1, \ldots, C_m). \]

The automaton \( C_m \) of Lemma 3.10 runs \( m \) copies of the domain automaton \( D \). Each copy reads one of the \( m \) infinite words over \( E \) extracted from the infinite input word over \( (E \times \{0, 1\})^m \). Automaton \( C_m \) rejects if one of these copies rejects.

Let \( T = (D, r, S_1, \ldots, S_n, R_1, \ldots, R_p) \) be a RA tree over the set of directions \( E = \{1, \ldots, n\} \) with \( T = (\mathcal{M}_D, \mathcal{M}_r, (\mathcal{M}_1)_{1 \leq i \leq n}, (\mathcal{M}_j)_{1 \leq j \leq p}) \) its canonical presentation, and let \( \varphi \) be a cMSO-formula\(^6\), with free variables \( X_1, \ldots, X_m \).

\[^6\] in the language defined by the grammar (1)
We define the set
\[ \text{enc}(\varphi^T) := \bigcup_{C_1, \ldots, C_m \in \text{Ch}(T)} \text{enc}(C_1, \ldots, C_m) \quad (2) \]

**Proposition 3.11** Given \( \varphi \) a \( c\text{MSO} \)-formula\(^7\) with free variables \( X_1, \ldots, X_m \), one can effectively construct a Büchi automaton \( B_{\varphi, T} \) that recognizes \( \text{enc}(\varphi^T) \).

We can now conclude the proof of Theorem 3.9. Observe that model checking against formula \( \varphi \) over the RA tree \( T \) amounts to deciding whether the language of the Büchi automaton \( B_{\varphi, T} \) of Proposition 3.11 is non-empty. Indeed, this language is not empty if, and only if, there is some assignment \( \sigma \) of the second order-variables \( X_1, \ldots, X_m \) such that \( T, \sigma \models \varphi \). In particular, if \( \varphi \) is closed (\( m = 0 \)), by convention we have:

- If \( T \not\models \varphi \), the union in Equation (2) is empty as it ranges over an empty set;
- If \( T \models \varphi \), the union in Equation (2) is not empty as it is equal to \( \text{enc}(\cdot) \).

Notice that Proposition 2.12 from [3] regarding the decidability of the full MSO-theory (and not only \( c\text{MSO} \)) for automatic presentations based on a singleton alphabet is now a direct corollary of Theorem 3.9 since every set of words over a singleton alphabet is a chain.

From Theorem 3.9 on the decidability of \( c\text{MSO} \) over RA trees, and because binary relations \( S^*, \preceq, \text{el} \) come for free with automata constructions (Lemma 3.7), we have the following.

**Corollary 3.12** Model checking against \( c\text{MSO}[r, S_1, \ldots, S_n, R_1, \ldots, R_p, S^*, \preceq, \text{el}, =] \) is decidable over a RA trees.

Notice that if we restrict to trees with only relations \( S_1 \ldots S_n \), model checking against the full logic MSO becomes decidable (Rabin’s Theorem [17]). However, considering extra relations (e.g. “equal level”) makes the model checking against MSO undecidable over RA trees, as we already saw in Theorem 2.13 of Section 2, which is not the case for \( c\text{MSO} \) by Corollary 3.12.

Also, since every path quantifier in a path-\( \text{MSO} \)-formula can be translated in \( c\text{MSO} \), and since considering extra arbitrary relations in trees, such as epistemic relations, does not harm this translation, we obtain the following result, which to our knowledge has not been considered in the literature.

**Corollary 3.13** Given a RA tree \( T = \langle D, r, S_1, \ldots, S_n, R_1, \ldots, R_p \rangle \) and a close path-\( \text{MSO} \)-formula \( \varphi \) over the signature of \( T \), it can be decided if \( T \models \varphi \).

We now turn to logical formalisms that combine knowledge and time in the spirit of [13].

---

\(^7\) with syntax of (1)
4 Logics of knowledge and time over RA trees

We now exploit the peculiarities of RA trees to consider extensions of several classic formalisms and study their expressivity. We first recall our criterion for expressivity.

**Definition 4.1** A logic $L$ is *embedded into* a logic $L'$, whenever there is an effective translation that maps every $L$-formula onto an equivalent $L'$-formula, namely for every regular automatic tree $T$, we have: $T \models \varphi$ if, and only if, $T \models \varphi'$.

Notice that when $L$ is embedded into $L'$, the decidability of the model checking against $L'$ entails the decidability of the model checking against $L$.

In tree structures, time is captured by the generalized successor binary relation $S := \bigcup_{i=1}^{n} S_i$, as in computation trees for temporal logics such as CTL [8], CTL* [12], etc. Depending on their arities, the other relations $R_1, \ldots, R_p$ have different roles. Unary relations label nodes with atomic propositions that range over some fixed set $AP = \{p, q, \ldots\}$. Binary relations, written $K_1 \ldots K_m$ as the epistemic modalities in multi-agent epistemic logic, will play the role of knowledge modalities as in classic epistemic logic. On this basis, we shortly write $cMSO_K$ for the logic $cMSO[r, S_1, \ldots, S_n, (p)_{p \in AP}, K_1 \ldots K_m]$.

The ability to quantify over chains allows us to capture properties along branches of the trees that are naturally stated in linear-time mu-calculus. Recall that MSO along linear orders, hence branches, is expressively equivalent to Büchi automata that capture all $\omega$-regular properties, as shown by [24], and so does the linear-time mu-calculus [9]. We therefore introduce the logic $BL_{\mu}^K$ that is based on the linear-time mu-calculus [4,27], equipped with knowledge modalities but also with second order quantifications over branches of the tree, just as CTL* extends LTL to the branching-time semantics.

The logic $BL_{\mu}^K$ is called the *branching epistemic linear-time mu-calculus*; in the spirit of CTL*, it relies on two kinds of formulas: state formulas ($\Phi$) and path formulas ($\varphi$).

**Definition 4.2** The syntax of $BL_{\mu}^K$ is as follows.

- **State formulas**: $\Phi ::= p \mid K_i \Phi \mid \neg \Phi \mid (\Phi \land \Phi) \mid E \varphi$ where $i \in \{1, \ldots, m\}$.
- **Path formulas**: $\varphi ::= Z \mid \Phi \mid \neg \varphi \mid (\varphi \land \varphi) \mid X \varphi \mid \mu Z. \varphi[Z]$ where $\varphi$ is closed in $E \varphi$, $Z \in V$ is under the scope of an even number of negations in $\varphi[Z]$.

State formula $K_i \Phi$ is read as ‘Agent $i$ knows $\Phi$’ and state formula $E \varphi$ is read as ‘there is a path starting from the current state that satisfies $\varphi$’. Path formula $X \varphi$ is read as ‘$\varphi$ holds in the next state’. Path formula $\mu Z. \varphi[Z]$ is the linear mu-calculus fix-point construction. State formula $A \varphi$ is an abbreviation of $\neg E \neg \varphi$ and is read as ‘$\varphi$ holds in all paths starting from the current state’.

All formulas of $BL_{\mu}^K$ are interpreted in a tree $T = \langle D, r, S_1, \ldots, S_n, K_1 \ldots K_m, (p)_{p \in AP} \rangle$. State formulas are interpreted on nodes, while path formulas are interpreted on branches.
• State formulas:
  \( \mathcal{T}, d \models p \) \iff \( d \in p \);
  \( \mathcal{T}, d \models K_i \Phi \) \iff \( \text{for all } d' \in D \text{ s.t. } (d, d') \in K_i \), we have \( \mathcal{T}, d' \models \Phi \);
  \( \mathcal{T}, d \models \neg \Phi \) \iff \( \mathcal{T}, d \models \Phi \);
  \( \mathcal{T}, d \models (\Phi \land \Psi) \) \iff \( \mathcal{T}, d \models \Phi \) and \( \mathcal{T}, d \models \Psi \);
  \( \mathcal{T}, d \models E\varphi \) \iff \( \text{there exists a maximal chain } C \text{ with least element } d \) (i.e. a branch starting from \( d \)) such that \( \mathcal{T}, C \models \varphi \);

• Path formulas: we add a valuation \( \sigma : V \to 2^N \) for the free second-order variables in the formulas, and write \( \sigma^+ \) for the valuation \( \sigma^+(Z) = \{ n | n + i \in \sigma(Z) \} \):
  \( \mathcal{T}, \pi, \sigma \models Z \) \iff \( 0 \in \sigma(Z) \);
  \( \mathcal{T}, \pi, \sigma \models \Phi \) \iff \( \mathcal{T}, \pi(0) \models \Phi \);
  \( \mathcal{T}, \pi, \sigma \models (\varphi \land \psi) \) \iff \( \mathcal{T}, \pi, \sigma \models \varphi \) and \( \mathcal{T}, \pi, \sigma \models \psi \);
  \( \mathcal{T}, \pi, \sigma \models \neg \varphi \) \iff \( \mathcal{T}, \pi, \sigma \models \varphi \);
  \( \mathcal{T}, \pi, \sigma \models \mu Z. \varphi[Z] \) \iff \( 0 \) is in the least-fix point of \( \|\varphi[Z]\|_\pi^\sigma \)

\[ \{ \|\varphi[Z]\|_\pi^\sigma : 2^N \to 2^N \}
\]

\[ \mathcal{P} \to \{ i \in \mathbb{N} \mid \pi(i) \pi(i + 1) \ldots (\sigma[Z \mapsto \mathcal{P}])^+ \models \varphi[Z] \} \].

A tree (resp. a forest) satisfies a state formula if it satisfies it from its root node (resp. from all its distinguished root nodes).

Notice that the least fix-point always exists. Indeed, the function \( \|\varphi[Z]\|_\pi^\sigma \) is monotone (recall that we have required that every variable to occur under the scope of an even number of negations) and the Knaster-Tarski Theorem [20] applies in the complete lattice \( (2^N, \subseteq) \). Besides, we can alternatively define the least fix-point of \( \|\varphi[Z]\|_\pi^\sigma \) as \( \bigcap \{ F \subseteq 2^N \ | \|\varphi[Z]\|_\pi^\sigma(F) \subseteq F \} \).

Example 4.3
• \( \mathcal{T}, \pi \models \mu Z. p \) if, and only if, \( \mathcal{T}, \pi(0) \models p \).
  • \( \mathcal{T}, \pi \models \mu Z. (XZ \lor p) \) if, and only if, there exists a node \( \pi(i) \) along \( \pi \) such that \( \mathcal{T}, \pi(i) \models p \).
  • \( \mathcal{T}, \pi \models \mu Z. (\psi \lor (\varphi \land XZ)) \) if, and only if, there exists a node \( \pi(j) \) such that \( \mathcal{T}, \pi(j) \models \psi \) and \( \mathcal{T}, \pi(i) \models \varphi \) for every \( 0 \leq i < j \).

By means of fix-points in \( \text{BL}_{\mu K} \), we capture the linear-time logic \( \text{LTL} \) [16], and since we also have the existential path quantifier \( \mathcal{E} \), we can embed the full logic \( \text{CTL}^* \) of [12] in \( \text{BL}_{\mu K} \).

Now the epistemic modalities \( K_i \) allow us to capture \( \text{CTL}^* \), the logic \( \text{CTL}^* \) equipped with knowledge modalities. More precisely, we introduce the following macro for ‘\( \varphi \) until \( \psi \)’: \( \varphi \mathcal{U} \psi \) is an abbreviation of formula \( \mu Z. (\psi \lor (\varphi \land XZ)) \).

Definition 4.4
\( \text{CTL}^* \) is the syntactic fragment of \( \text{BL}_{\mu K} \) defined by:

• State formulas: \( \Phi ::= \Phi \land \Phi | \neg \Phi | E \varphi | K_i \Phi | p \) where \( i \in \{1, \ldots, m \} \).

• Path formulas: \( \varphi ::= \Phi | \neg \varphi | \varphi \land \varphi | X \varphi | \varphi \mathcal{U} \varphi \).

The syntactic fragment \( \text{CTL}^* \) of \( \text{CTL}^* \) has only state formulas:
\[ \Phi ::= \Phi \land \Phi | \neg \Phi | EX \varphi | E(\Phi \mathcal{U} \Phi) | A(\Phi \mathcal{U} \Phi) | K_i \Phi | p \]
**Proposition 4.5** \(\text{CTLK} \) is embedded into \(\text{FO}[K_1 \ldots K_m, (p)_{p \in \text{AP}}, S, S^*, \preceq, =].\)

We end this section with the main Theorem 4.6 that states the decidability of model checking against \(B_{\mu}^{\text{lin}}K\) over RA trees. First of all, it should be clear that the following holds.

**Theorem 4.6** \(B_{\mu}^{\text{lin}}K\) is embedded into \(\text{cMSO}^K\).

**Corollary 4.7** Model checking against \(B_{\mu}^{\text{lin}}K\) is decidable over automatic regular trees.

Figure 4 is a recap of the expressivity results we have obtained (arrows resulting from transitivity are omitted), where an arrow \(\mathcal{L} \rightarrow \mathcal{L}'\) means that \(\mathcal{L}\) is embedded into \(\mathcal{L}'\).

### 5 Application to epistemic planning synthesis

As announced, we show that Theorem 3.9 together with Proposition 3.11 have an interesting impact in the domain of epistemic planning. The epistemic planning setting as introduced by \[6\] relies on Dynamic epistemic logic, that we recall here. Next we explain how the original problem can be generalized to attain more expressive planning goals, such as statements (i)–(iii) of the introduction, while maintaining the decidability frontier for free. Our results generalize the ones obtained in \[15,1\], but make great use of the already established property \[15, Lemma 22, p. 109\] that, under the assumption that preconditions and postconditions of events are propositional, the relational structure that contains all plan candidates is automatic, and actually it is a RA tree (see Theorem 5.5).

#### 5.1 Preliminaries on Dynamic epistemic logic

As earlier, \(\text{AP}\) denotes the set of atomic propositions with typical elements \(p, q, \ldots\), and \(A_g\) is finite set of agents with typical elements \(a, b, \ldots\). The propositional language is denoted by \(\mathcal{L}_{\text{Prop}}\) and the language of multi-agent epistemic modal logic is denoted by \(\mathcal{L}_{\text{EL}}\).

**Definition 5.1** A Kripke model \(\mathcal{M} = (W, (K_a)_{a \in A_g}, V)\) is defined by a non-empty set \(W\) of epistemic worlds, epistemic relations \((K_a)_{a \in A_g} \subseteq W \times W\) and a valuation function \(V : W \rightarrow 2^{\text{AP}}\).

A pair \((\mathcal{M}, w)\) is called a pointed epistemic model.

Figure 5 bottom left shows a pointed epistemic model.

The dynamic of the system is captured by \textit{event models}. An event model is like a Kripke model but possible worlds are instead events equipped with a
precondition and a postcondition. We intentionally denote the set of events $E$ with the same notation that the set of directions in trees (Definition 2.2), as they will denote directions in the DEL structure (see Definition 5.4).

**Definition 5.2** An event model $E = (E, (K^E_a)_{a \in Ag}, pre, post)$ is defined by a non-empty set of events $E$, epistemic relations $(K^E_a)_{a \in Ag} \subseteq E \times E$, a precondition function $pre: E \rightarrow \mathcal{L}_{EL}$ and a postcondition function $post: E \times AP \rightarrow \mathcal{L}_{Prop}$. When $pre: E \rightarrow \mathcal{L}_{Prop}$, the event model $E$ is propositional.

A pair $(E, e)$ is called a pointed event model, where $e$ represents the actual event.

**Definition 5.3** Let $\mathcal{M} = (W, (K^\mathcal{M}_a)_{a \in Ag}, V)$ be a Kripke model. Let $\mathcal{E} = (E, (K^\mathcal{E}_a)_{a \in Ag}, pre, post)$ be an event model. The product of $\mathcal{M}$ and $\mathcal{E}$ is $\mathcal{M} \otimes \mathcal{E} = (W', (K^\mathcal{M}'_a, V'))$ where:

- $W' = \{(w, e) \in W \times E \mid \mathcal{M}, w \models pre(e)\}$;
- $((w, e), (w', e')) \in K^\mathcal{M}'_a$ iff $(w, w') \in K^\mathcal{M}_a$ and $(e, e') \in K^\mathcal{E}_a$;
- $V'((w, e)) = \{p \in AP \mid \mathcal{M}, w \models post(e, p)\}$.

Figure 5 top shows an example of a pointed event model with two events $e$ and $f$. The actual event is $e$ but agent $b$ imagines event $f$ as the sole possible event. Also, the product is given in bottom right.

### 5.2 DEL structures

In this subsection, following [15], we incorporate the initial epistemic model $\mathcal{M}$ and the infinitely many the products $\mathcal{M} \otimes \mathcal{E}^n := \mathcal{M} \otimes \mathcal{E} \otimes \cdots \otimes \mathcal{E}$ into a single structure, called a DEL structure, denoted by $\mathcal{M} \mathcal{E}^\ast$. Worlds of $\mathcal{M} \otimes \mathcal{E}^n$, called histories, are naturally denoted by words of the form $w_1 \ldots e_n$ where $w$ is a world of $\mathcal{M}$ and $e_1, \ldots, e_n$ are events of $\mathcal{E}$. E.g. the world $((w, e_1), e_2)$ is denoted $we_1e_2$. The pair $(\mathcal{M}, \mathcal{E})$ is called the DEL presentation of $\mathcal{M} \mathcal{E}^\ast$. A DEL presentation $(\mathcal{M}, \mathcal{E})$ is propositional if $\mathcal{E}$ is propositional.
Definition 5.4 [15, Def. 54, p. 105] The DEL structure denoted by \((\mathcal{M}, \mathcal{E})\) is the structure \(\mathcal{M}\mathcal{E}^* = (\mathcal{H}, (S_e)_{e \in \mathcal{E}}, (K_a)_{a \in \mathcal{A}_g}, (p)_{p \in \mathcal{AP}})\), where \(\mathcal{H}\) is the disjoint union of the sets of worlds of \(\mathcal{M} \otimes \mathcal{E}^n\) – namely, the histories; \((h, h') \in S_e\) whenever \(e\); \((h, h') \in K_a\) whenever \(h\) and \(h'\) are worlds of \(\mathcal{M} \otimes \mathcal{E}^n\), for some \(n\), and \((h, h') \in K'_a\) where \(K'_a\) is the epistemic relation for agent \(a\) in \(\mathcal{M} \otimes \mathcal{E}^n\); and finally, \(h \in p\) whenever \(p \in V(h)\) in \(\mathcal{M} \otimes \mathcal{E}^n\).

From the proof of [15, Lemma 22, p. 109], one can easily show that:

Theorem 5.5 Propositional DEL structures \(\mathcal{M}\mathcal{E}^*\) are finite sets (forests) of RA trees, and their presentation is effectively computable from \((\mathcal{M}, \mathcal{E})\).

We now turn to the epistemic planning problems.

5.3 Generalized epistemic planning and plan synthesis

The epistemic planning problem, as originally stated by [6] is defined as follows.

Definition 5.6 Epistemic planning problem

Input: a pointed epistemic model \(\mathcal{M}, w\), an event model \(\mathcal{E}\), an \(\mathcal{L}_{EL}\)-formula \(\psi\);

Output: Yes, if there is a sequence of events \(e_1, \ldots, e_n\) in \(\mathcal{E}\) such that \(\mathcal{M}\mathcal{E}^n, we_1 \ldots e_n \models \psi\).

We generalize this epistemic planning problem by considering a model checking problem against \(BL_{\mu}K\) over DEL structures.

Definition 5.7 Generalized epistemic planning problem

Input: a pointed epistemic model \(\mathcal{M}, w\), an event model \(\mathcal{E}\), an \(BL_{\mu}K\)-formula \(\varphi\);

Output: Yes, if \(\mathcal{M}\mathcal{E}^* \models \varphi\).

The problem of Definition 5.7 is indeed a generalization of epistemic planning problem of Definition 5.6 in the sense that the latter can be reduced to the former by model checking the formula \(\varphi := EF\psi\), where \(\psi\) is the planning goal.

The following example shows the relevance of generalized epistemic planning.

Example 5.8 We give \(BL_{\mu}K\)-formulas \(\varphi\) for the three statements (i)–(iii) of the introduction.

(i) ‘invariantly, an intruder \(a\) does not know the location of the piece of jewelry more than 3 consecutive steps’ can be expressed as \(EG(\Theta \rightarrow (X \neg \Theta \vee X^2 \neg \Theta \vee X^3 \neg \Theta)\) where \(\Theta := \bigvee_{\ell \in \text{Loc}} K_a\text{pieceOfJewelleryIn}(\ell)\).

(ii) ‘all drones know that the region is safe every 20 steps’ can be expressed as \(E\nu Z.(\bigwedge_{a \in \mathcal{A}_g} K_a\text{regionSafe} \land X^{20}Z)\).

(iii) ‘with the current plan, the drone \(a\) never knows the region is safe but every 10 steps, there is a(nother) plan to let the drone \(a\) eventually know the region is safe’ can be expressed as \(EG[\neg K_a\text{regionSafe} \land \nu Z.(EF K_a\text{regionSafe} \land X^{10}Z)]\).
The following result subsumes the decidability result for the epistemic planning problem for propositional DEL presentations ([30], [15]), and is a mere corollary of Theorems 5.5, 4.6 and 3.9.

**Theorem 5.9** The generalized epistemic planning problem is decidable for propositional DEL presentations.

In other terms, in the literature, decidability was established for reachability goals and Theorem 5.9 says that the problem remains decidable for goals that are arbitrary $\text{BL}^{\text{lin}}_{\mu}K$-formulas $\varphi$. However, when considering planning, only formulas $\varphi$ of the form $E\varphi'$ are relevant. For such formulas, we can effectively build an automaton that recognizes exactly all plans achieving $\varphi'$. This automaton arises from the automata constructions for $\text{BL}^{\text{lin}}_{\mu}K$-formulas, made possible by the automata constructions for $\text{cMSO}$-formulas, and the fact that every $\text{BL}^{\text{lin}}_{\mu}K$-formula can be effectively translated into a $\text{cMSO}$-formula. The algorithm to obtain this “plan automaton” is as follows.

(i) Compute the $\text{cMSO}$-formula $\chi$ equivalent to $\varphi'$ (Th. 4.6). By the translation of $\text{BL}^{\text{lin}}_{\mu}K$ into $\text{cMSO}$, formula $\chi$ has a single free second-order variable $X$, interpreted as a path, i.e. a plan;

(ii) Compute $B_\chi$ (Proposition 3.11) which accepts all paths achieving goal $\chi$, or equivalently $\varphi'$.

Actually, automaton $B_\chi$ reads infinite words over alphabet $E \times \{0, 1\}$, where the second component of the letters is always equal to 1, since the specified chain is required to be a path. Projecting a word accepted by $B_\chi$ onto the first components of its letters provides a plan.

**6 Discussion**

We have adapted the seminal result of the decidability of the FO-theory of every automatic structure by augmenting the logic up to $\text{cMSO}$, a logic relying on chain-MSO of [12, 21], but by restricting to the subclass of RA trees. A nice application of this result is the decidability of the generalized epistemic planning problem for propositional DEL presentations.

Regarding the computational complexity of our algorithms, it should be observed that the automata construction to model check against $\text{cMSO}$ (Proposition 3.11) is non elementary in the number of alternations between existential and universal quantifiers in the formula (each underlying negation yields an exponentiation blow-up for the complementation). Still, it would be relevant to investigate in practice, if the automata have particular shapes that do not reach this non-elementary worst-case upper-bound complexity.

Compared to the work of Maubert [15] with $\text{CTL}^*K$, we offer the entire logic $\text{cMSO}$ (or $\text{BL}^{\text{lin}}_{\mu}K$) that strictly subsumes $\text{CTL}^*K$, and yet allowing to solve the epistemic planning problems over the same class of planning domains (i.e. propositional DEL presentations). However, it should be noticed that [15] also considers a different problem called the *epistemic protocol synthesis problem*, which differs from the generalized epistemic planning problem: in the
epistemic protocol synthesis problem, one has to prune the tree structure so that the remaining satisfies a $\text{CTL}^*\text{K}$-goal. This subject is out of the scope of this paper, but suggests some comments.

The existence of a pruning, i.e. a sub-tree, that satisfies the protocol goal, requires the use of second-order quantifiers ranging over arbitrary subsets of nodes, that is the full logic $\text{MSO}$ which cannot be model checked (Theorem 2.13). It is an open question whether there is a logic in between c$\text{MSO}$ and full $\text{MSO}$ to solve the epistemic protocol synthesis problem with $\text{BL}^\text{lin}_\mu\text{K}$ specifications. The existence of a pruning of the tree that satisfies the protocol goal, requires the use of second-order quantifiers ranging over arbitrary subsets of nodes, that is the full logic $\text{MSO}$ which cannot be model checked (Theorem 2.13). It is an open question whether or not there is room for a logic in between c$\text{MSO}$ and full $\text{MSO}$ to solve the epistemic protocol synthesis problem with $\text{BL}^\text{lin}_\mu\text{K}$ specifications. Also, the synthesis technique deployed by [15] actually applies to a family of tree structures that is larger than the one of RA trees. At the moment, the technique does not adapt to logics which involve arbitrary fix-points. It is an open question whether or not the work of [15] can be faithfully extended to deal the entire alternation-free fragment of $\text{BL}^\text{lin}_\mu\text{K}$.

Acknowledgments We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

References

20 Chain-MSO Logic over Regular Automatic Trees and Epistemic Planning Synthesis


