Model-Checking for Ability-Based Logics with Constrained Plans

Stéphane Demri, Raul Fervari

Model-Checking for Ability-Based Logics with Constrained Plans

Stéphane Demri[1], **Raul Fervari**[2,3]

[1]LMF, CNRS, ENS Paris-Saclay, University Paris-Saclay, France
[2]FAMAF, Universidad Nacional de Córdoba & CONICET, Argentina
[3]Guangdong Technion - Israel Institute of Technology, China

Abstract

We investigate the complexity of the model-checking problem for a family of modal logics capturing the notion of "knowing how". We consider the most standard ability-based knowing how logic, for which we show that model-checking is PSpace-complete. By contrast, a multi-agent variant based on an uncertainty relation between plans in which uncertainty is encoded by a regular language, is shown to admit a PTime model-checking problem. We extend with budgets the above-mentioned ability-logics, as done for ATL-like logics. We show that for the former logic enriched with budgets, the complexity increases to at least ExpSpace-hardness, whereas for the latter, the PTime bound is preserved. Other variant logics are discussed along the paper.

1 Introduction

Knowing How Logics. The epistemic concept of "knowing how" has received considerable attention lately, as a new way of providing formal foundations to strategic reasoning in AI and automated planning (see e.g., (van Ditmarsch et al. 2015)). Most formalisations for this notion are based on simple combinations of standard knowledge modalities and abilities (Herzig and Troquard 2006; van der Hoek and Lomuscio 2003). However, arguably, such an approach does not lead to a proper characterisation of "knowing how", as discussed in (Herzig 2015; Jamroga and Ågotnes 2007).

In (Wang 2018b), a novel proposal to reason about "knowing how" assertions was introduced. The logic (herein written $\mathcal{L}_{kh}$) includes a goal-direct knowing how modality $\mathsf{Kh}(\mathtt{p},\mathtt{q})$, indicating that *"the agent knows how to achieve the goal* $\mathtt{q}$*, whenever the initial condition* $\mathtt{p}$ *holds"*. As stated in (Wang 2018b), there is an explicit intention to keep the logical language neat and to introduce simple semantical structures, namely labelled directed graphs. The interpretation of the new modality is *ability-based*: the formula $\mathsf{Kh}(\mathtt{p},\mathtt{q})$ holds whenever there is a plan σ (understood as a finite sequence of actions) such that from all the states satisfying $\mathtt{p}$, the execution of σ leads only to states satisfying $\mathtt{q}$; and any partial execution of σ can be extended to a complete execution of σ (known as the strong executability condition, directly inspired from conformant planning (Smith and Weld 1998)). Thus, "knowing how" is given by the abilities described by the graph. The simplicity of the logical language is partly reflected by the fact that formulae of the form $\mathsf{Kh}(\mathtt{p},\mathtt{q})$ are global, no action symbol appears in formulae, a single agent is considered, and no "knowing that" modality is present. A complete axiomatisation is provided in (Wang 2018b) but more importantly, such a work has been a source of inspiration for many others. Some variants include: multiple agents, other classes of plans, or admit "knowing that" operators (see e.g. (Fervari et al. 2017; Li and Wang 2021b)). Other approaches, related to strategic games and coalitions, have been studied in (Naumov and Tao 2018c; Naumov and Tao 2018a; Naumov and Tao 2018b). Finally, the logic studied in (Areces et al. 2021) (called herein $\mathcal{L}^{\mathrm{U}}$) is based on a notion of indistinguishability over plans. Arguably, such a proposal provides a more epistemic view of knowing how than other approaches.

Substantial progress has been already done related to philosophical motivations, axiom systems and combinations with other epistemic operators. However, much less contributions exist about the possibility to constraint plans (for instance, by adding budgets), or to perform the model-checking task (instead of checking theoremhood). These issues are addressed in this paper.

Our Motivations. The need to investigate knowing how logics with constrained plans is advocated in (Li 2017, Ch. 3). This is our first motivation to revisit results about numerical constraints (i.e. budgets) and/or regularity constraints about plans, over this family of logics. Moreover, it is more natural to investigate model-checking instead of the validity/satisfiability problem, not only because it is a standard problem in formal verification, but also because the power of adding constraints is best illustrated there. This is due to the fact that the available plans are part of the input, and a few clever tricks cannot be used anymore, such as guessing simply shaped plans witnessing the truth of a formula, see e.g. the completeness proofs in (Li 2017; Areces et al. 2021). Also, most of the works are dedicated to Hilbert-style axiomatisation and to the satisfiability problem, and relatively little was done on model-checking, apart from the works (Li, Yu, and Wang 2017; Areces et al. 2021; Li and Wang 2021a). Since the semantics of such logics involve complex conditions such as strong executability, PTime upper bounds seemed more difficult to reach than what is known about the minimal modal logic K

or alternating-time temporal logic ATL (known to admit a PTime model-checking problem, see e.g. (Alur, Henzinger, and Kupferman 2002)). An additional motivation to study the model-checking problem for ability-based logics rests on consequences for the satisfiability problem. It is known that when a logic has the (bounded) small model property, a decision procedure for checking satisfiability consists in guessing a small model and then perform a model-checking test. This is done in (Li 2017, Th. 3.4.8) to decide the satisfiability status of formulae in an extension of $\mathcal{L}_{kh}$, in which the satisfaction of Kh-formulae requires the satisfaction of constraints for intermediate states. Then, knowing the complexity of the model-checking problem becomes crucial to evaluate the complexity of the satisfiability problem. This is the case in (Areces et al. 2021, Sec. 3.3), where the satisfiability problem for an uncertainty-based knowing how logic is shown in NP by stating that model-checking is in PTime.

Our Contributions. We study the model-checking problem for known ability-based logics from the literature, typically from (Wang 2015) and (Li 2017, Ch. 3), but also for extensions by adding arithmetical and/or regular constraints on plans. Furthermore, we provide new relationships of this family of logics with formal languages and automata theory.

As a warm-up, we show that the model-checking problem for the logic of knowing how from (Wang 2015) (written $\mathcal{L}_{kh}$) is PSpace-complete (Thm. 1). This untractability result came a bit as a surprise, but witnesses the high expressive power of the Kh-like modalities despite its simplicity. A variant of the logic $\mathcal{L}_{kh}$ designed in (Areces et al. 2021) and featuring a notion of indistinguishable plans, has been shown therein to admit a PTime model-checking problem. Here, we assume that each of those indistinguishability classes is specified by a regular language, thus, can be potentially infinite. We prove that the model-checking problem for this extension $\mathcal{L}^{\mathrm{U}}_{reg}$ can be also solved in PTime, using an algorithm based on reachability checks (Thm. 2).

We extend the logics $\mathcal{L}_{kh}$ and $\mathcal{L}^{\mathrm{U}}_{reg}$ so that actions have costs, and executing a plan requires to stay within a certain budget. Adding budget-like constraints to plans is advocated in (Li and Wang 2017; Cao and Naumov 2017). The extensions we propose follow a standard pattern already used in ATL-like logics (see e.g. (Alechina et al. 2017; Alechina et al. 2018; Bulling and Goranko 2022)), in energy games (see e.g. (Bouyer et al. 2008)) and in general for other transition systems (see e.g. (Cao and Naumov 2017)). We show that the PTime upper bound is preserved if budgets are added to $\mathcal{L}^{\mathrm{U}}_{reg}$ (written $\mathcal{L}^{\mathrm{U}}_{reg}(\star)$, Thm. 3). This is done by taking advantage of algorithms for solving decision problems related to vector addition systems with states (VASS) (see e.g. (Karp and Miller 1969)), restricted to a single counter, and using Bellman-Ford algorithm for weighted directed graphs (see e.g. (Cormen et al. 2022)). By constrast, we show that adding budgets to $\mathcal{L}_{kh}$ increases the complexity significantly. The model-checking problem for $\mathcal{L}_{kh}(\star)$ (with resources) is ExpSpace-hard (Lemma 7). The subproblem in which the cost of any action does not depend on the state it is triggered, is shown ExpSpace-complete (Thm. 4).

2 Preliminaries on a Simple Knowing How Logic

We present below the knowing how logic introduced in (Wang 2015; Wang 2018b) (herein written $\mathcal{L}_{kh}$). Formulae of the form $\mathsf{Kh}(\varphi, \psi)$ can be read as *"when φ is the case, the agent knows how to make ψ true"*. Since $\mathcal{L}_{kh}$ has a central position in this paper, we dedicate a separate section for preliminary definitions, and for its model-checking problem. Notions introduced here, such as LTS, plans and strong executability, will be helpful in forthcoming sections.

Formulae and models. Let Prop be a countably infinite set of propositional symbols. **Formulae of the logic $\mathcal{L}_{kh}$** are defined by the grammar:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{Kh}(\varphi, \varphi) \qquad (\mathsf{p} \in \mathsf{Prop}).$$

Other Boolean connectives are defined as usual.

In (Wang 2015; Wang 2018b), formulae are interpreted over *labelled transition systems* (LTSs): relational models in which each (basic) relation indicates the source and target of a particular type of action the agent can perform. Let Act be a countably infinite set of action symbols. A **labelled transition system (LTS)** is a tuple $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathsf{a}})_{\mathsf{a}\in\mathsf{Act}}, \mathrm{V})$, where:

- S is a non-empty set of states (called the domain);
- $(\mathrm{R}_{\mathsf{a}})_{\mathsf{a}\in\mathsf{Act}}$ is a collection of binary relations on S;
- $\mathrm{V} : \mathrm{S} \to 2^{\mathsf{Prop}}$ is a labelling function.

Given an LTS $\mathcal{S}$ and s a state, the pair $(\mathcal{S}, s)$ (parentheses usually dropped) is called a **pointed LTS**. Before defining the satisfaction relation for the logic $\mathcal{L}_{kh}$, we need to introduce the notion of linear plan, that is used all along the paper (for more complex plans, see (Li and Wang 2021b, Def. 18)).

Linear plans. Let Act^* be the set of finite sequences over Act. Elements of Act^* are called **plans**, with ε being the **empty plan**. Given $\sigma \in \mathsf{Act}^*$, let $|\sigma|$ be the length of σ ($|\varepsilon| \stackrel{\text{def}}{=} 0$). For $0 \leqslant k \leqslant |\sigma|$, the plan σ_k is σ's initial segment up to (and including) the kth position (with $\sigma_0 \stackrel{\text{def}}{=} \varepsilon$). For $0 < k \leqslant |\sigma|$, the action $\sigma[k]$ is the one in σ's kth position.

Let $(\mathrm{R}_{\mathsf{a}})_{\mathsf{a}\in\mathsf{Act}}$ be a family of binary relations with each $\mathrm{R}_{\mathsf{a}} \subseteq \mathrm{S}\times\mathrm{S}$. Define $\mathrm{R}_{\varepsilon} \stackrel{\text{def}}{=} \{(s,s) \mid s \in \mathrm{S}\}$ and, for all $\sigma \in \mathsf{Act}^*$ and $\mathsf{a} \in \mathsf{Act}$, $\mathrm{R}_{\sigma\mathsf{a}} \stackrel{\text{def}}{=} \{(s,t) \in \mathrm{S}\times\mathrm{S} \mid \exists\, t' \in \mathrm{S} \text{ s.t. } (s,t') \in \mathrm{R}_{\sigma} \text{ and } (t',t) \in \mathrm{R}_{\mathsf{a}}\}$. Take a plan $\sigma \in \mathsf{Act}^*$: for $s \in \mathrm{S}$ define $\mathrm{R}_{\sigma}(s) \stackrel{\text{def}}{=} \{t \in \mathrm{S} \mid (s,t) \in \mathrm{R}_{\sigma}\}$. For all $X \subseteq \mathrm{S}$ and $\Pi \subseteq \mathsf{Act}^*$, define $\mathrm{R}_{\Pi}(X) \stackrel{\text{def}}{=} \bigcup_{s\in X, \sigma\in\Pi} \mathrm{R}_{\sigma}(s)$.

Intuitively, (Wang 2015; Wang 2018a; Wang 2018b) handle how to express that an agent knows how to achieve ψ given φ, when she has an appropriate plan that allows her to go from any state in which φ holds only to states in which ψ holds. In order to characterise what 'appropriate' means, one can impose restrictions on what qualifies as an adequate plan.

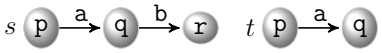

Figure 1: Example of LTS.

Let $(\mathrm{R}_{\mathtt{a}})_{\mathtt{a}\in\mathsf{Act}}$ as before. A plan $\sigma \in \mathsf{Act}^*$ is **strongly executable** (SE) at $s \in \mathrm{S}$ iff for all $k \in [0, |\sigma| - 1]$ and $t \in \mathrm{R}_{\sigma_k}(s)$, we have $\mathrm{R}_{\sigma[k+1]}(t) \neq \varnothing$. We define the set $\mathrm{SE}(\sigma) \stackrel{\text{def}}{=} \{s \in \mathrm{S} \mid \sigma \text{ is SE at } s\}$. Interestingly, in the next section we show that for any state s, the set $\{\sigma \in \mathsf{Act}^* \mid s \in \mathrm{SE}(\sigma)\}$ is a regular language.

Satisfaction relation ($\Vdash$). Let $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathtt{a}})_{\mathtt{a}\in\mathsf{Act}}, \mathrm{V})$ be an LTS, $s \in \mathrm{S}$ and φ be a formula of $\mathcal{L}_{kh}$, $\Vdash$ is defined as:

$$
\begin{array}{lcl}
\mathcal{S}, s \Vdash \mathtt{p} & \stackrel{\text{def}}{\Leftrightarrow} & \mathtt{p} \in \mathrm{V}(s), \\
\mathcal{S}, s \Vdash \neg\varphi & \stackrel{\text{def}}{\Leftrightarrow} & \mathcal{S}, s \nVdash \varphi, \\
\mathcal{S}, s \Vdash \varphi \vee \psi & \stackrel{\text{def}}{\Leftrightarrow} & \mathcal{S}, s \Vdash \varphi \text{ or } \mathcal{S}, s \Vdash \psi, \\
\mathcal{S}, s \Vdash \mathsf{Kh}(\varphi, \psi) & \stackrel{\text{def}}{\Leftrightarrow} & \text{there exists } \sigma \in \mathsf{Act}^* \text{ such that} \\
 & & (1)\ [\![\varphi]\!]^{\mathcal{S}} \subseteq \mathrm{SE}(\sigma) \text{ and} \\
 & & (2)\ \mathrm{R}_{\sigma}([\![\varphi]\!]^{\mathcal{S}}) \subseteq [\![\psi]\!]^{\mathcal{S}},
\end{array}
$$

with $[\![\chi]\!]^{\mathcal{S}} \stackrel{\text{def}}{=} \{s \in \mathrm{S} \mid \mathcal{S}, s \Vdash \chi\}$. If a plan $\sigma \in \mathsf{Act}^*$ satisfies conditions (1) and (2) above, we say that σ **witnesses** the satisfaction of $\mathcal{S}, s \Vdash \mathsf{Kh}(\varphi, \psi)$. In the LTS $\mathcal{S}$ of Fig. 1, we have that $\mathcal{S}, s \Vdash \mathsf{Kh}(\mathtt{p}, \mathtt{q})$ (via the single-action plan $\mathtt{a}$) whereas $\mathcal{S}, s \nVdash \mathsf{Kh}(\mathtt{p}, \mathtt{r})$, as the plan $\mathtt{ab}$ is not strongly executable at $t \in [\![\mathtt{p}]\!]^{\mathcal{S}}$.

A complete Hilbert-style system for $\mathcal{L}_{kh}$ can be found in (Wang 2015) (see also (Wang 2018b, Sec. 3)). Decidability of the satisfiability problem for $\mathcal{L}_{kh}$ is shown in (Li 2017, Ch. 3) (the proof is done for a more general logic), but its complexity characterisation is open, as far as we know.

3 Model-checking Problem for $\mathcal{L}_{kh}$

An LTS $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathtt{a}})_{\mathtt{a}\in\mathsf{Act}}, \mathrm{V})$ is **finite** iff S, Act and Prop are finite sets. The **model-checking problem for the logic** $\mathcal{L}_{kh}$, written $\mathrm{MC}(\mathcal{L}_{kh})$, is defined as follows.

Input: a finite LTS $\mathcal{S}$ and a formula φ over Prop, $s \in \mathrm{S}$.

Question: $\mathcal{S}, s \Vdash \varphi$?

Though the model-checking problem for many modal and temporal logics can be solved in PTime, see e.g. (Blackburn, de Rijke, and Venema 2001; Demri, Goranko, and Lange 2016), below, we show that $\mathrm{MC}(\mathcal{L}_{kh})$ is PSpace-complete, requiring more computational resources. This high complexity was probably not expected from the works in the literature. Indeed, the model-checking problem is either not considered explicitly at all, or shown in PTime for some variant knowing how logics (see e.g. (Li and Wang 2021a; Li and Wang 2021b)) or shown PSpace-hard but for some expressive logic, see e.g. (Li, Yu, and Wang 2017).

PSpace-completeness of $\mathrm{MC}(\mathcal{L}_{kh})$. To show PSpace-hardness, we reduce the nonemptiness problem for intersection of deterministic finite-state automata that is a PSpace-complete problem, see e.g. (Galil 1976; Kozen 1977).

Lemma 1. *$\mathrm{MC}(\mathcal{L}_{kh})$ is PSpace-hard.*

Proof. (sketch) Given $N \in \mathbb{N}$, for each $i \in [1, N]$, let $\mathcal{A}_i = (Q_i, \mathsf{Act}, \delta_i, q_i, F_i)$ be a deterministic finite-state automaton accepting the regular language $\mathrm{L}(\mathcal{A}_i) \subseteq \mathsf{Act}^*$. We assume that each automaton $\mathcal{A}_i$ is complete and the sets of locations are disjoint. We define an LTS $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathtt{a}})_{\mathtt{a}\in\mathsf{Act}}, \mathrm{V})$ over the set $\{\mathtt{init}, \mathtt{fin}\} \subseteq \mathsf{Prop}$, with $\mathrm{S} \stackrel{\text{def}}{=} \biguplus Q_i$ such that $\mathcal{S}, q_1 \Vdash \mathsf{Kh}(\mathtt{init}, \mathtt{fin})$ iff $\mathrm{L}(\mathcal{A}_1) \cap \cdots \cap \mathrm{L}(\mathcal{A}_N) \neq \varnothing$ (q_1 is arbitrary), as follows.

- For all $q, q' \in \mathrm{S}$ and $\mathtt{a} \in \mathsf{Act}$, $(q, q') \in \mathrm{R}_{\mathtt{a}} \stackrel{\text{def}}{\Leftrightarrow}$ there is $i \in [1, N]$ such that $q, q' \in Q_i$ and $q \xrightarrow{\mathtt{a}} q' \in \delta_i$.
- For all $i \in [1, N]$ and $q \in Q_i$, $\mathtt{init} \in \mathrm{V}(q) \stackrel{\text{def}}{\Leftrightarrow} q = q_i$, and $\mathtt{fin} \in \mathrm{V}(q) \stackrel{\text{def}}{\Leftrightarrow} q \in F_i$.

$\mathrm{MC}(\mathcal{L}_{kh})$ is PSpace-hard, since $\mathsf{Kh}(\mathtt{init}, \mathtt{fin})$ and $\mathcal{S}$ are computed in logspace in the size of the input automata. □

In order to establish that $\mathrm{MC}(\mathcal{L}_{kh})$ is in PSpace, we show a small plan property based on the regular structure of the set of plans witnessing the satisfaction of $\mathcal{S}, s \Vdash \mathsf{Kh}(\varphi_1, \varphi_2)$. By regularity, we mean that the set of plans can be shown to be a regular language, and we can *effectively* compute a finite-state automaton accepting the language of witness plans. Hence, though the regularity property is instrumental to prove our PSpace upper bound, we believe it is also interesting for its own sake to understand the expressive power of the modality Kh in $\mathcal{L}_{kh}$. Below, we state a few properties before getting to the PSpace upper bound.

Given an LTS $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathtt{a}})_{\mathtt{a}\in\mathsf{Act}}, \mathrm{V})$, let us characterise the plans $\sigma \in \mathsf{Act}^*$ such that either (non1) $[\![\varphi_1]\!]^{\mathcal{S}} \nsubseteq \mathrm{SE}(\sigma)$ or (non2) $\mathrm{R}_{\sigma}([\![\varphi_1]\!]^{\mathcal{S}}) \nsubseteq [\![\varphi_2]\!]^{\mathcal{S}}$. Let $\mathcal{A}_{(t_1,t_2)} = (Q, \mathsf{Act}, \delta, I, F)$ be the automaton defined as follows.

- $Q \stackrel{\text{def}}{=} \mathrm{S}$, $I \stackrel{\text{def}}{=} \{t_1\}$, $F \stackrel{\text{def}}{=} \{t_2\}$.
- For all $t, t' \in Q$ and $\mathtt{a} \in \mathsf{Act}$, $t \xrightarrow{\mathtt{a}} t' \in \delta \stackrel{\text{def}}{\Leftrightarrow} (t, t') \in \mathrm{R}_{\mathtt{a}}$.

It is worth observing that $t_2 \in \mathrm{R}_{\sigma}(t_1)$ iff $\sigma \in \mathrm{L}(\mathcal{A}_{(t_1,t_2)})$. So, the plans σ satisfying (non2) are exactly those in

$$\bigcup \{\mathrm{L}(\mathcal{A}_{(t_1,t_2)}) \mid t_1 \in [\![\varphi_1]\!]^{\mathcal{S}}, t_2 \in [\![\neg\varphi_2]\!]^{\mathcal{S}}\},$$

defined from at most $|\mathrm{S}|^2$ languages. We write $\bar{\mathcal{A}}_{(t_1,t_2)}$ to denote the powerset automaton built from $\mathcal{A}_{(t_1,t_2)}$ such that $\mathrm{L}(\bar{\mathcal{A}}_{(t_1,t_2)}) = \mathsf{Act}^* \backslash \mathrm{L}(\mathcal{A}_{(t_1,t_2)})$ (complement language). Hence, σ satisfies (2) iff it belongs to the language below.

$$\bigcap \{\mathrm{L}(\bar{\mathcal{A}}_{(t_1,t_2)}) \mid t_1 \in [\![\varphi_1]\!]^{\mathcal{S}}, t_2 \in [\![\neg\varphi_2]\!]^{\mathcal{S}}\}.$$

The locations of the automaton $\bar{\mathcal{A}}_{(t_1,t_2)}$ are subsets of S.

Let us handle now the condition (non1). We have (non1) iff there is $t_1 \in [\![\varphi_1]\!]^{\mathcal{S}}$ such that $t_1 \notin \mathrm{SE}(\sigma)$, i.e. for some $k \in [0, |\sigma| - 1]$ and $u \in \mathrm{R}_{\sigma_k}(t_1)$, $\mathrm{R}_{\sigma[k+1]}(u) = \varnothing$. Equivalently, (non1) iff there are $X_1, X_2 \subseteq \mathrm{S}$ such that for some $k \in [0, |\sigma| - 1]$, $X_1 = \mathrm{R}_{\sigma_k}(t_1)$, $X_2 = \mathrm{R}_{\sigma_{k+1}}(t_1)$ and there is $u \in X_1$ such that for no $u' \in X_2$, we have

$(u, u') \in \mathrm{R}_{\sigma[k+1]}$. This characterisation serves as the basis to define the set $\{\sigma \mid s \in \mathrm{SE}(\sigma)\}$ for some state $s \in \mathrm{S}$. Let $\mathcal{A}^\star_s = (Q, \mathsf{Act}, \delta, I, F)$ be the finite-state automaton defined as follows.

- $Q \stackrel{\text{def}}{=} \mathcal{P}(\mathrm{S}) \times \{acc, rej\}$, $I \stackrel{\text{def}}{=} \{(\{s\}, acc)\}$, $F \stackrel{\text{def}}{=} \mathcal{P}(\mathrm{S}) \times \{acc\}$ ('acc' stands for 'acceptance', 'rej' for 'rejection').
- For all $X \in \mathcal{P}(\mathrm{S})$ and $\mathtt{a} \in \mathsf{Act}$, $(X, rej) \xrightarrow{\mathtt{a}} (X, rej) \in \delta$.
- For all $X, X' \in \mathcal{P}(\mathrm{S})$ and $\mathtt{a} \in \mathsf{Act}$, $(X, acc) \xrightarrow{\mathtt{a}} (X', acc) \in \delta \stackrel{\text{def}}{\Leftrightarrow}$ $(\star)$ $\mathrm{R}_{\mathtt{a}}(X) = X'$ and $(\star\star)$ for all $t \in X$, there is $t' \in X'$ such that $(t, t') \in \mathrm{R}_{\mathtt{a}}$.
- For all $X, X' \in \mathcal{P}(\mathrm{S})$ and $\mathtt{a} \in \mathsf{Act}$, $(X, acc) \xrightarrow{\mathtt{a}} (X', rej) \in \delta \stackrel{\text{def}}{\Leftrightarrow}$ $(\star)$ and not $(\star\star)$.

Lemma 2. $\{\sigma \mid s \in \mathrm{SE}(\sigma)\} = \mathrm{L}(\mathcal{A}^\star_s)$.

Thus, the plan σ satisfies (1) iff it belongs to the language

$$\bigcap \{\mathrm{L}(\mathcal{A}^\star_t) \mid t \in [\![\varphi_1]\!]^{\mathcal{S}}\},$$

defined from at most $|\mathrm{S}|$ languages; moreover, each automaton $\mathcal{A}^\star_t$ has at most $2 \cdot 2^{|\mathrm{S}|}$ locations. In conclusion, the set of plans witnessing $\mathcal{S}, s \Vdash \mathsf{Kh}(\varphi_1, \varphi_2)$ is equal to

$$\bigcap \{\mathrm{L}(\mathcal{A}^\star_t) \mid t \in [\![\varphi_1]\!]^{\mathcal{S}}\} \cap \bigcap \{\mathrm{L}(\bar{\mathcal{A}}_{(t_1,t_2)}) \mid t_1 \in [\![\varphi_1]\!]^{\mathcal{S}}, t_2 \in [\![\neg\varphi_2]\!]^{\mathcal{S}}\},$$

which amounts to build an automaton with at most $2^{|\mathrm{S}|} \cdot |\mathrm{S}|^2 \cdot 2^{2 \cdot |\mathrm{S}|} \cdot |\mathrm{S}|$ locations.

As we have provided a way of characterising conditions (1) and (2), we can state an interesting property about the plans witnessing the satisfaction of $\mathcal{S}, s \Vdash \mathsf{Kh}(\varphi_1, \varphi_2)$. This property has been unnoticed so far, as far as we can judge.

Corollary 1. *Let $\mathcal{S}$ be an LTS, s be a state and $\mathsf{Kh}(\varphi_1, \varphi_2)$ be a formula. Let L be set of plans witnessing $\mathcal{S}, s \Vdash \mathsf{Kh}(\varphi_1, \varphi_2)$. Then, L is a regular language and if $\mathrm{L} \neq \varnothing$, then it contains a plan of length at most $2^{3 \cdot |\mathrm{S}|} \cdot |\mathrm{S}|^3$.*

We state the key property to get the PSpace upper bound.

Lemma 3. *Checking whether $\mathcal{S}, s \Vdash \mathsf{Kh}(\mathtt{p}, \mathtt{q})$ (with $\mathtt{p}, \mathtt{q} \in \mathsf{Prop}$) can be done in polynomial space.*

Proof. (sketch) To test whether $\mathcal{S}, s \Vdash \mathsf{Kh}(\mathtt{p}, \mathtt{q})$, we check on-the-fly the non-emptiness of the product automaton

$$\times \{\mathcal{A}^\star_t \mid t \in [\![\mathtt{p}]\!]^{\mathcal{S}}\} \times \times \{\bar{\mathcal{A}}_{(t_1,t_2)} \mid t_1 \in [\![\mathtt{p}]\!]^{\mathcal{S}}, t_2 \in [\![\neg\mathtt{q}]\!]^{\mathcal{S}}\}.$$

There are at most $|\mathrm{S}| + |\mathrm{S}|^2$ automata and each location from those automata is a subset of S, possibly enriched with a flag either acc or rej. Polynomial space is enough to store two consecutive locations of the product automaton, and its transition relation can be also computed in polynomial space based on the LTS $\mathcal{S}$. Nonemptiness of finite-state automata can be checked in NLogSpace (the number of states of the product automaton is bounded by $2^{3 \cdot |\mathrm{S}|} \cdot |\mathrm{S}|^3$). So, we get a NPSpace decision procedure for checking nonemptiness of the product automaton. By Savitch's Theorem (Savitch 1970), we get the PSpace upper bound. □

It is time to characterise the complexity of $\mathrm{MC}(\mathcal{L}_{kh})$.

Theorem 1. $\mathrm{MC}(\mathcal{L}_{kh})$ *is* PSpace*-complete.*

PSpace-hardness is a direct consequence of Lemma 1. In order to establish PSpace-easiness, we can design a standard labelling algorithm taking advantage of Lemma 3 (this idea can be adapted to all the logics studied in this paper).

Variant problems. Our approach using formal languages theory provides us with modular and standard tools in order to characterise the complexity of model-checking problems for several ability-based logics. This is the case for instance, of the logic EPDL, known to be PSpace-complete (Li, Yu, and Wang 2017, Th. 4.3), for which our results can be easily adapted. On the other hand, alternative constraints on plans can be considered. For instance, a modality $\mathsf{Kh}_{\mathcal{A}}$ (parametric on the finite-state automaton $\mathcal{A}$) such that a witness plan must necessarily belong to $\mathrm{L}(\mathcal{A})$. Another option is to consider variant logics from the literature, such as the logic $\mathcal{L}_{khm}$ from (Wang 2018b, Sec. 4) and (Li 2017, Ch. 3)). $\mathcal{L}_{khm}$ contains a ternary modality $\mathsf{Kh}^m(\varphi_1, \varphi_2, \varphi_3)$, and witness plans must satisfy φ_2 all along the traversed paths. It is not difficult to see that we can apply our techniques for all these logics (details are omitted due to lack of space).

4 Regular Classes in Uncertainty-Based Logics

It is argued in (Areces et al. 2021) that an epistemic notion of knowing how should be not only based on the given abilities, but also on some notion of indistinguishability/uncertainty between them. Therein, it is also argued that such an approach makes the framework closer to standard epistemic logics (van Ditmarsch et al. 2015). For instance, it makes easier to move to a multi-agent setting.

In the *uncertainty-based* setting, agents share the same set of *affordances* (provided by the actual environment, typically an LTS). Still, they have different *abilities* depending on which of these affordances are available for each of them, and how well they can tell these affordances apart.

In this section, we assume that the indistinguishability classes of plans are specified by regular languages and we prove that the model-checking problem for this extension $\mathcal{L}^{\mathrm{U}}_{reg}$ is also in PTime.

4.1 Uncertainty-Based Knowing How

Let Agt be a finite set of agent symbols. **Formulae of the logic $\mathcal{L}^{\mathrm{U}}$** are defined by the grammar:

$$\varphi ::= \mathtt{p} \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{Kh}_{\mathtt{a}}(\varphi, \varphi) \qquad (\mathtt{p} \in \mathsf{Prop}, \mathtt{a} \in \mathsf{Agt}).$$

A formula $\mathsf{Kh}_{\mathtt{a}}(\varphi, \psi)$ is read as *"when φ is the case, the agent $\mathtt{a}$ knows how to make ψ true"*.

Here, each agent's knowledge is not given simply by the abilities described by the LTS. Instead, the models are enriched with an **uncertainty relation** $\sim_{\mathtt{a}}$ for each agent $\mathtt{a}$, which is an equivalence relation over a non-empty subset $\Pi_{\mathtt{a}} \subseteq \mathsf{Act}^*$ describing those plans that are indistinguishable from each other, from the agent's perspective. Below, each relation $\sim_{\mathtt{a}}$ is represented by its set of equivalence classes $\mathrm{U}_{\mathtt{a}}$ (therefore, $\mathrm{U}_{\mathtt{a}}$ satisfies a few simple properties recalled below). A **multi-agent uncertainty-based LTS (LTS$^{\mathrm{U}}$)** is a tuple $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathtt{a}})_{\mathtt{a} \in \mathsf{Act}}, (\mathrm{U}_{\mathtt{a}})_{\mathtt{a} \in \mathsf{Agt}}, \mathrm{V})$ where

$(\mathrm{S}, (\mathrm{R})_{\mathsf{a}\in\mathsf{Act}}, \mathrm{V})$ is an LTS and each $\mathrm{U}_\mathfrak{a}$ assigns to the agent $\mathfrak{a}$ a non-empty collection of pairwise disjoint non-empty sets of plans with: (1) $\mathrm{U}_\mathfrak{a} \neq \varnothing$, (2) $\varnothing \notin \mathrm{U}_\mathfrak{a}$, and

(3) $\Pi_1, \Pi_2 \in \mathrm{U}_\mathfrak{a}$ with $\Pi_1 \neq \Pi_2$ implies $\Pi_1 \cap \Pi_2 = \varnothing$.

Intuitively, $\Pi_\mathfrak{a} \stackrel{\text{def}}{=} \bigcup_{\Pi\in\mathrm{U}_\mathfrak{a}} \Pi$ is the set of plans that the agent $\mathfrak{a}$ has at her disposal. Similarly, as in classical epistemic logic, $\sim_\mathfrak{a} \subseteq \Pi_\mathfrak{a} \times \Pi_\mathfrak{a}$ describes agent $\mathfrak{a}$'s indistinguishability. This relation is not defined over possible states of affairs, but rather over her available plans.

Again, in this setting we need to define the notion of being a "proper plan". Let $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_\mathsf{a})_{\mathsf{a}\in\mathsf{Act}}, (\mathrm{U}_\mathfrak{a})_{\mathfrak{a}\in\mathsf{Agt}}, \mathrm{V})$ be an LTS$^\mathrm{U}$. For $\Pi \subseteq \mathsf{Act}^*$ and $X \cup \{s\} \subseteq \mathrm{S}$, we define $\mathrm{R}_\Pi \stackrel{\text{def}}{=} \bigcup_{\sigma\in\Pi} \mathrm{R}_\sigma$, $\mathrm{R}_\Pi(s) \stackrel{\text{def}}{=} \bigcup_{\sigma\in\Pi} \mathrm{R}_\sigma(s)$, and $\mathrm{R}_\Pi(X) \stackrel{\text{def}}{=} \bigcup_{t\in X} \mathrm{R}_\Pi(t)$. A set of plans $\Pi \subseteq \mathsf{Act}^*$ is **strongly executable** at $s \in \mathrm{S}$ iff every plan $\sigma \in \Pi$ is strongly executable at s. Hence, $\mathrm{SE}(\Pi) \stackrel{\text{def}}{=} \bigcap_{\sigma\in\Pi} \mathrm{SE}(\sigma)$ is the set of the states in S where Π is strongly executable.

The notion of satisfiability for $\mathsf{Kh}_\mathfrak{a}$-formulae is defined as:

$$\mathcal{S}, s \Vdash \mathsf{Kh}_\mathfrak{a}(\varphi, \psi) \stackrel{\text{def}}{\Leftrightarrow} \text{there exists } \Pi \in \mathrm{U}_\mathfrak{a} \text{ such that}$$
$$(1)\ [\![\varphi]\!]^\mathcal{S} \subseteq \mathrm{SE}(\Pi) \ \text{ and } \ (2)\ \mathrm{R}_\Pi([\![\varphi]\!]^\mathcal{S}) \subseteq [\![\psi]\!]^\mathcal{S}.$$

Let $\mathcal{S}$ be the LTS from Fig. 1, enriched with $\mathrm{U}_\mathfrak{a} = \{\{\mathsf{a}, \mathsf{ab}\}\}$ and $\mathrm{U}_\mathfrak{b} = \{\{\mathsf{a}\}, \{\mathsf{ab}\}\}$, for some $\mathfrak{a}, \mathfrak{b} \in \mathsf{Agt}$. With this setup, the agent $\mathfrak{a}$ cannot distinguish between the plans a and ab (i.e., she is uncertain about whether these plans lead to the same outcome or not), whereas the agent $\mathfrak{b}$ considers them different. Thus, $\mathcal{S}, s \Vdash \neg\mathsf{Kh}_\mathfrak{a}(\mathsf{p}, \mathsf{q}) \wedge \mathsf{Kh}_\mathfrak{b}(\mathsf{p}, \mathsf{q})$.

The satisfiability problem for $\mathcal{L}^\mathrm{U}$ is NP-complete, whereas $\mathrm{MC}(\mathcal{L}^\mathrm{U})$ is in PTime (Areces et al. 2021). In an instance of $\mathrm{MC}(\mathcal{L}^\mathrm{U})$, each $\mathrm{U}_\mathfrak{a}$ is defined such that $\Pi_\mathfrak{a}$ is finite. Representing equivalence relations by their indistinguishability classes results handy for designing model-checking algorithms, as we can directly deal with the list of available plans. However, there is a limitation: we only consider a finite set of them. For instance, assuming that $\mathrm{L}(\mathsf{a}^*\mathsf{b}) \subseteq \Pi_\mathfrak{a}$, we may wish to express that for all $\sigma \neq \sigma' \in \mathrm{L}(\mathsf{a}^*\mathsf{b})$, we have $\sigma \sim_\mathfrak{a} \sigma'$ (the action a is silent for the agent $\mathfrak{a}$). In a more concrete example, we can consider that an agent $\mathfrak{a}$ is a server sending connection requests to another server. The indistinguishability class for $\mathfrak{a}$ is given by $\mathsf{aa}^\star$, where action a stands for "sending a connection request through a secure channel". Thus, the class establishes that after the first request is sent by agent $\mathfrak{a}$, all other requests are ignored. This type of class cannot be handled when $\mathrm{U}_\mathfrak{a}$ is a finite set of *finite* sets of plans. Instead, below, the elements of $\mathrm{U}_\mathfrak{a}$ are defined as finite-state automata $\mathcal{A}$ such that $\mathrm{L}(\mathcal{A})$ is understood as a (possibly infinite) equivalence class. This is a standard way to represent finitely a (potential) infinite set of words.

4.2 Generalisation with Regular Equivalence Classes

Strictly speaking, we generalise the problem $\mathrm{MC}(\mathcal{L}^\mathrm{U})$ from (Areces et al. 2021), by requiring elements in $\mathrm{U}_\mathfrak{a}$ to be regular languages (instead of finite sets), but defined through finite-state automata. On the logical side, the models for the logic $\mathcal{L}^\mathrm{U}_{reg}$ defined below, are a subclass of the models for $\mathcal{L}^\mathrm{U}$ (because we assume regularity of each equivalence class). Hence, the introduction of the logic $\mathcal{L}^\mathrm{U}_{reg}$ serves at least two purposes: to generalise $\mathrm{MC}(\mathcal{L}^\mathrm{U})$ and to introduce models in which the indistinguishability classes are regular languages, which might seem confusing at first glance.

An **LTS$^\mathrm{U}$ with regular constraints (reg-LTS$^\mathrm{U}$)** is a tuple $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_\mathsf{a})_{\mathsf{a}\in\mathsf{Act}}, (\mathrm{U}_\mathfrak{a})_{\mathfrak{a}\in\mathsf{Agt}}, \mathrm{V})$ where each $\mathrm{U}_\mathfrak{a}$ assigns to agent $\mathfrak{a}$ a non-empty collection of finite-state automata $\varnothing \neq \mathrm{U}_\mathfrak{a} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$, such that each $\mathcal{A} \in \mathrm{U}_\mathfrak{a}$ defines an equivalence class over a set $\Pi_\mathfrak{a} = \bigcup_{\mathcal{A}\in\mathrm{U}_\mathfrak{a}} \mathrm{L}(\mathcal{A}) \subseteq 2^{\mathsf{Act}^*}$; i.e., for all $\mathcal{A}_j, \mathcal{A}_k \in \mathrm{U}_\mathfrak{a}$, $j \neq k$ implies $\mathrm{L}(\mathcal{A}_j) \cap \mathrm{L}(\mathcal{A}_k) = \varnothing$.

The clause for $\mathsf{Kh}_\mathfrak{a}$-formulae over reg-LTS$^\mathrm{U}$s becomes:

$$\mathcal{S}, s \Vdash \mathsf{Kh}_\mathfrak{a}(\varphi, \psi) \stackrel{\text{def}}{\Leftrightarrow} \text{there is } \mathcal{A} \in \mathrm{U}_\mathfrak{a} \text{ such that}$$
$$(1)\ [\![\varphi]\!]^\mathcal{S} \subseteq \mathrm{SE}(\mathrm{L}(\mathcal{A})) \ \text{ and } \ (2)\ \mathrm{R}_{\mathrm{L}(\mathcal{A})}([\![\varphi]\!]^\mathcal{S}) \subseteq [\![\psi]\!]^\mathcal{S}.$$

Unlike (Areces et al. 2021), each equivalence class over $\Pi_\mathfrak{a}$ is defined by a regular language. Interestingly, theoremhood in $\mathcal{L}^\mathrm{U}$ and $\mathcal{L}^\mathrm{U}_{reg}$ coincide (see (Areces et al. 2021, Th. 2)).

A reg-LTS$^\mathrm{U}$ $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_\mathsf{a})_{\mathsf{a}\in\mathsf{Act}}, (\mathrm{U}_\mathfrak{a})_{\mathfrak{a}\in\mathsf{Agt}}, \mathrm{V})$ is **finite** iff S, Act, Prop and the $\mathrm{U}_\mathfrak{a}$'s are finite sets. The **model-checking problem for the logic** $\mathcal{L}^\mathrm{U}_{reg}$, written $\mathrm{MC}(\mathcal{L}^\mathrm{U}_{reg})$, is defined over finite reg-LTS$^\mathrm{U}$.

In (Areces et al. 2021) it has been shown that the model checking problem for $\mathcal{L}^\mathrm{U}$ can be solved in PTime. However, herein the main difficulty is dealing with automata generating potentially infinite languages, and thus, equivalence classes with infinitely many plans. One can show that the method used in the previous section applies also here, but giving us (roughly) an ExpSpace upper bound. However, as we will show here, this bound is not optimal. We shall use a more fine-tuned approach to handle the conditions related to the modality $\mathsf{Kh}_\mathfrak{a}$. With this at hand, we will show that $\mathrm{MC}(\mathcal{L}^\mathrm{U}_{reg})$ belongs actually to PTime.

The semantics of $\mathsf{Kh}_\mathfrak{a}(\varphi, \psi)$ requires to find some $\mathcal{A} \in \mathrm{U}_\mathfrak{a}$, such that $\mathrm{L}(\mathcal{A})$ satisfies two conditions. As $\mathrm{U}_\mathfrak{a}$ is finite, inspecting its members one by one, poses no difficulties. Again our method relies on characterising all the potential candidates for the satisfiability of a $\mathsf{Kh}_\mathfrak{a}$-formula that fail to be a proper witness. For $\mathcal{A} \in \mathrm{U}_\mathfrak{a}$, we proceed as follows.

1. Checking condition (2) $\mathrm{R}_{\mathrm{L}(\mathcal{A})}([\![\varphi]\!]^\mathcal{S}) \subseteq [\![\psi]\!]^\mathcal{S}$ can be handled by checking that
$$\mathrm{L}(\mathcal{A}) \cap \bigcup\{\mathrm{L}(\mathcal{A}_{(t_1,t_2)}) \mid t_1 \in [\![\varphi]\!]^\mathcal{S}, t_2 \in [\![\neg\psi]\!]^\mathcal{S}\} = \varnothing,$$
with $\mathcal{A}_{(t_1,t_2)}$ defined as in the previous section. By the analysis provided there, and since nonemptiness of the intersection of *two* finite-state automata can be checked in PTime, we have a PTime procedure for checking (2).

2. Then, we will design a PTime algorithm for checking the strong executability of $\mathrm{L}(\mathcal{A})$ at $[\![\varphi]\!]^\mathcal{S}$. The algorithm relies on constructing a directed graph as a product between $\mathcal{A}$ and $\mathcal{S}$. Then, we proceed by looking for a state t that fails to execute a "productive" action, i.e., an action that leads to a final state of $\mathcal{A}$, and checking whether (q, t) is reachable, for some location q that is accepting in $\mathcal{A}$. For the sake of correctness, we assume that all the locations in $\mathcal{A}$ are productive, i.e. for any location, there is a path in $\mathcal{A}$ leading to a final location.

Let $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathtt{a}})_{\mathtt{a}\in\mathsf{Act}}, (\mathrm{U}_{\mathfrak{a}})_{\mathfrak{a}\in\mathsf{Agt}}, \mathrm{V})$ be a reg-LTS$^{\mathrm{U}}$ and $\mathcal{A} = (Q, \mathsf{Act}, \delta, I, F)$ be an automaton. We define a digraph $\mathcal{G} = (V, E)$ such that $V \stackrel{\text{def}}{=} Q \times \mathrm{S}$ and for all $(q,t), (q',t') \in V$, we have $(q,t) \to (q',t') \in E \stackrel{\text{def}}{\Leftrightarrow}$ there is some $\mathtt{a} \in \mathsf{Act}$ such that $q \xrightarrow{\mathtt{a}} q' \in \delta$ and $(t,t') \in \mathrm{R}_{\mathtt{a}}$. $\mathcal{G}$ is therefore a (standard) product between $\mathcal{A}$ and $\mathcal{S}$. Given $s \in \mathrm{S}$, we define the procedure $\mathtt{CheckSE}(\mathcal{S}, s, \mathcal{A})$ as follows.

1. For each $\mathtt{a} \in \mathsf{Act}$, for each $t \in \mathrm{S}$ such that $\mathrm{R}_{\mathtt{a}}(t) = \varnothing$ and for each $q \in Q$ such that $\delta(q, \mathtt{a}) \neq \varnothing$ do:
   (a) if there is $q_0 \in I$ such that there is a path from (q_0, s) to (q,t) in $\mathcal{G}$, then: return True;
2. return False.

Lemma 4. *$s \notin \mathrm{SE}(\mathrm{L}(\mathcal{A}))$ iff $\mathtt{CheckSE}(\mathcal{S}, s, \mathcal{A})$ returns True.*

Now it is time to establish the complexity bound.

Theorem 2. *$\mathrm{MC}(\mathcal{L}^{\mathrm{U}}_{reg})$ is in* PTime.

Proof. (sketch) PTime is guaranteed if $\mathcal{S}, s \Vdash \mathsf{Kh}_{\mathfrak{a}}(\varphi, \psi)$ can be checked in PTime. $\mathrm{U}_{\mathfrak{a}}$ contains a linear amount of finite-state automata and we showed that (2) can be checked in PTime. It remains to verify that so is the case for (1). By Lemma 4, (1) holds iff for all $s' \in [\![\varphi]\!]^{\mathcal{S}}$, $\mathtt{CheckSE}(\mathcal{S}, s', \mathcal{A})$ returns False. As $|[\![\varphi]\!]^{\mathcal{S}}| \leqslant |\mathrm{S}|$ and the for loop ranges over all S, Act and Q, checking (a) is performed $\mathcal{O}(|\mathrm{S}|^2 \cdot |\mathsf{Act}| \cdot |Q|)$ times, which is polynomial in the size of the instance $\mathcal{S}, s \Vdash \mathsf{Kh}_{\mathfrak{a}}(\varphi, \psi)$. Moreover, (a) is an instance of the graph accessibility problem (GAP), known to be NLogSpace-complete, applied on the graph $\mathcal{G}$ of quadratic size in the size of $\mathcal{S}$. Thus, (1) can be checked in PTime. □

5 Ability-Based Logics with Budgets

The need to express budget-like constraints about plans has been advocated in (Li 2017, Sec. 3.1) and (Li and Wang 2017). Assuming that actions have costs, the execution of plans requires that the agent stays always within the budget. Adding resource reasoning is a well-known paradigm used in ATL-like logics, see e.g. (Alechina et al. 2017), in energy games, see e.g. (Chatterjee, Doyen, and Henzinger 2017), and in multi-agent systems, see e.g. (Cao and Naumov 2017). Herein, we study the complexity of adding resource reasoning in the logics $\mathcal{L}_{kh}$ and $\mathcal{L}^{\mathrm{U}}_{reg}$, respectively. It is particularly interesting to observe is that the worst-case complexity stays within PTime for the extension of $\mathcal{L}^{\mathrm{U}}_{reg}$ whereas it jumps to at least ExpSpace-hardness for $\mathcal{L}_{kh}$. We provide clues to understand this computational gap.

Given a set of states S and a finite set of actions Act, a **weight function** is a map $wf : \mathrm{S} \times \mathsf{Act} \to \mathbb{Z}^r$ for some $r \geqslant 0$; $wf(s, \mathtt{a})$ is understood as the cost of executing the action $\mathtt{a}$ at state s. Adding wf to an LTS shall be our standard way to enrich models with action costs. Given a computation $\lambda = s_0 \xrightarrow{\mathtt{a}_1} s_1 \xrightarrow{\mathtt{a}_2} s_2 \cdots \xrightarrow{\mathtt{a}_K} s_K$, its weight is defined as $wf(\lambda) \stackrel{\text{def}}{=} \Sigma_{k=1}^{K} wf(s_{k-1}, \mathtt{a}_k)$ (empty computations have zero cost).

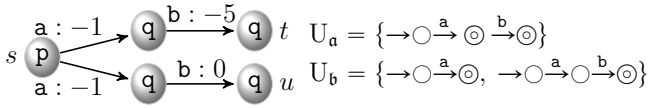

Figure 2: A reg-LTS$^{\mathrm{U}}$ with budgets ($r = 1$).

5.1 PTime Upper Bound for Model-checking $\mathcal{L}^{\mathrm{U}}_{reg}(\star)$

We write $\mathcal{L}^{\mathrm{U}}_{reg}(r)$ to denote the ability-based logic $\mathcal{L}^{\mathrm{U}}_{reg}$ augmented with $r \geqslant 0$ resource types, and $\mathcal{L}^{\mathrm{U}}_{reg}(\star)$ to denote the version with an arbitrary number of resource types. This is the version used in our model-checking problem.

For $r \geqslant 0$, the set of $\mathcal{L}^{\mathrm{U}}_{reg}(r)$ **formulae** is defined below.

$$\varphi ::= \mathtt{p} \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{Kh}^{\vec{b}}_{\mathfrak{a}}(\varphi, \varphi) \quad (\mathtt{p} \in \mathsf{Prop}, \mathfrak{a} \in \mathsf{Agt}, \vec{b} \in \mathbb{N}^r)$$

All the integers appearing in formulae and models are encoded with a binary representation. Formulae of the form $\mathsf{Kh}^{\vec{b}}_{\mathfrak{a}}(\varphi, \psi)$ are read as *"when φ is the case, the agent $\mathfrak{a}$ knows how to make ψ true with budget $\vec{b}$"*. Models of the logic $\mathcal{L}^{\mathrm{U}}_{reg}(r)$ are of the form $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathtt{a}})_{\mathtt{a}\in\mathsf{Act}}, (\mathrm{U}_{\mathfrak{a}})_{\mathfrak{a}\in\mathsf{Agt}}, wf, \mathrm{V})$, where $wf : \mathrm{S} \times \mathsf{Act} \to \mathbb{Z}^r$ is a weight function. A plan $\sigma = \mathtt{a}_1 \cdots \mathtt{a}_K$ is **$\vec{b}$-compatible** at s ($\vec{b} \in \mathbb{N}^r$) $\stackrel{\text{def}}{\Leftrightarrow}$ for every computation $\lambda = s_0 \xrightarrow{\mathtt{a}_1} s_1 \cdots \xrightarrow{\mathtt{a}_K} s_K$ with $s_0 = s$, we have for all $L \in [1, K]$, $\vec{b} + wf(\lambda_{\leqslant L}) \geqslant \vec{0}$ (with $\lambda_{\leqslant L} \stackrel{\text{def}}{=} s_0 \xrightarrow{\mathtt{a}_1} s_1 \cdots \xrightarrow{\mathtt{a}_L} s_L$). The plan σ is **$\vec{b}$-compatible** at a set $X \subseteq \mathrm{S} \stackrel{\text{def}}{\Leftrightarrow}$ it is $\vec{b}$-compatible at all $s \in X$. $\vec{b}$ is understood as the **initial budget**.

Given a model $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathtt{a}})_{\mathtt{a}\in\mathsf{Act}}, (\mathrm{U}_{\mathfrak{a}})_{\mathfrak{a}\in\mathsf{Agt}}, wf, \mathrm{V})$ and $s \in \mathrm{S}$, we update the satisfiability clause for $\mathsf{Kh}^{\vec{b}}_{\mathfrak{a}}$-formulae:

$$\mathcal{S}, s \Vdash \mathsf{Kh}^{\vec{b}}_{\mathfrak{a}}(\varphi, \psi) \stackrel{\text{def}}{\Leftrightarrow} \text{there is } \mathcal{A} \in \mathrm{U}_{\mathfrak{a}} \text{ such that}$$
$$(1)\ [\![\varphi]\!]^{\mathcal{M}} \subseteq \mathrm{SE}(\mathrm{L}(\mathcal{A})),\ \ (2)\ \mathrm{R}_{\mathrm{L}(\mathcal{A})}([\![\varphi]\!]^{\mathcal{M}}) \subseteq [\![\psi]\!]^{\mathcal{M}},$$
$$\text{and } (3) \text{ for all } \sigma \in \mathrm{L}(\mathcal{A}),\ \sigma \text{ is } \vec{b}\text{-compatible at } [\![\varphi]\!]^{\mathcal{M}}.$$

Consider the model $\mathcal{S}$ of Fig. 2 (we use standard notations for finite-state automata). Both plans $\mathtt{a}$ and $\mathtt{ab}$ lead to $\mathtt{q}$ states, but the cost of executing $\mathtt{ab}$ from s to u is -1, whereas from s to t is -6. Thus, $\mathcal{S}, s \Vdash \mathsf{Kh}^{5}_{\mathfrak{b}}(\mathtt{p}, \mathtt{q}) \wedge \neg\mathsf{Kh}^{5}_{\mathfrak{a}}(\mathtt{p}, \mathtt{q})$.

In the budget-free logic $\mathcal{L}^{\mathrm{U}}_{reg}$, the clause related to the satisfaction of $\mathsf{Kh}_{\mathfrak{a}}(\varphi, \psi)$ uses exactly (1) and (2) above. We have shown that such conditions can be checked in PTime. To prove that $\mathrm{MC}(\mathcal{L}^{\mathrm{U}}_{reg}(\star))$ is in PTime too, we establish that (3) can be checked in PTime for a given $\mathcal{A}$. Since $|\mathrm{S}|$ is less than the size of $\mathcal{S}$ and $\mathrm{MC}(\mathcal{L}^{\mathrm{U}}_{reg}(\star))$ can be solved by a standard type of labelling algorithm, it is sufficient to show that given $\mathcal{S}$, $t \in \mathrm{S}$ and a finite-state automaton $\mathcal{A}$, one can check in PTime that for all $\sigma \in \mathrm{L}(\mathcal{A})$, σ is $\vec{b}$-compatible at t. This is the purpose of the rest of this subsection.

Interestingly, the models for $\mathcal{L}^{\mathrm{U}}_{reg}(r)$ can be viewed as an extension of the vector addition systems with states (VASS) (Karp and Miller 1969), since in both models the transitions are labelled by tuples in $\mathbb{Z}^r$ encoding an update function. Below, we recall a few standard definitions about

VASS that are helpful in the sequel to characterise the complexity of $\mathrm{MC}(\mathcal{L}^{\mathrm{U}}_{reg}(\star))$.

A **vector addition system with states (VASS)** is a structure $\mathcal{V}{=}(Q, r, R)$, where Q is a finite set of **locations**, $r{\in}\mathbb{N}$ is its **dimension**, and R is a finite set of **transitions** in $Q \times \mathbb{Z}^r \times Q$. A **configuration** (resp. **pseudo-configuration**) in a VASS $\mathcal{V}$ is a pair $(q, \vec{x}) \in Q \times \mathbb{N}^r$ (resp. in $Q \times \mathbb{Z}^r$). Given pseudo-configurations $(q, \vec{x})$, $(q', \vec{x'})$ and a transition $T = q \xrightarrow{\vec{u}} q'$, we write $(q, \vec{x}) \xrightarrow{T} (q', \vec{x'})$ whenever $\vec{x'} = \vec{u} + \vec{x}$. A **pseudo-run** is defined as a sequence $\rho = (q_0, \vec{x}_0) \xrightarrow{T_1} (q_1, \vec{x}_1) \xrightarrow{T_2} (q_2, \vec{x}_2) \cdots$ of pseudo-configurations, where $(q_0, \vec{x}_0)$ is the **initial** pseudo-configuration. A **run** is a pseudo-run in which only configurations in $Q \times \mathbb{N}^r$ occur.

An r-VASS is a VASS with $r \geqslant 0$ counters. We begin by presenting a simple problem called **NONSAFE(VASS)**, strongly related to the non-satisfaction of the condition (3).

Input: a VASS $\mathcal{V}$ and a configuration $(q_0, \vec{x}_0) \in Q \times \mathbb{N}^r$.

Question: is there a finite pseudo-run $\rho = (q_0, \vec{x}_0) \rightarrow (q_1, \vec{x}_1) \rightarrow \cdots \rightarrow (q_n, \vec{x}_n)$ such that $\vec{x}_n \notin \mathbb{N}^r$? (is it possible to reach a negative value from $(q_0, \vec{x}_0)$?).

Lemma 5. *NONSAFE(VASS) is in* PTime.

Lemma 5 follows from the fact that one instance of NONSAFE(VASS) can be reduced to r instances of NONSAFE(1-VASS) (restriction to 1-VASS). The second step consists in showing that checking whether a negative value can be reached in a 1-VASS from a given initial configuration can be solved using Bellman-Ford algorithm working on weighted directed graphs.

Assume that $\mathcal{A} = (Q, \mathsf{Act}, \delta, I, F)$ and all the locations in Q are productive, so any run reaching a given location from some initial location can be completed as an accepting run.

One more step is needed to establish that (3) can be solved in PTime. Namely, we build a VASS $\mathcal{V} = (Q', r, R')$ with $Q' \stackrel{\text{def}}{=} \mathrm{S} \times Q$ such that not (3) iff there is $q_0 \in I$ and $t \in [\![\varphi]\!]^{\mathcal{S}}$ such that $\mathcal{V}$, $((t, q_0), \vec{b})$ is a positive instance of NONSAFE(VASS). It remains to define R'. We have $(s, q) \xrightarrow{\vec{u}} (s', q') \in R' \stackrel{\text{def}}{\Leftrightarrow}$ for some $\mathtt{a} \in \mathsf{Act}$, $(s, s') \in \mathrm{R}_{\mathtt{a}}$ and $q \xrightarrow{\mathtt{a}} q' \in \delta$ (synchronisation on actions) with $\vec{u} = \mathit{wf}(s, \mathtt{a})$.

Lemma 6. *There are $t \in [\![\varphi_1]\!]^{\mathcal{S}}$ and $\sigma \in \mathrm{L}(\mathcal{A})$ such that for some $j \in [1, |\sigma|]$, σ_j is not $\vec{b}$-compatible at t iff there are $t \in [\![\varphi_1]\!]^{\mathcal{S}}$ and $q_0 \in I$ such that $\mathcal{V}$, $((t, q_0), \vec{b})$ is a positive instance of NONSAFE(VASS).*

Now, we are in position to state our best result as far as a PTime model-checking problem is concerned.

Theorem 3. $\mathrm{MC}(\mathcal{L}^{\mathrm{U}}_{reg}(\star))$ *is in* PTime.

Let us briefly provide the argument to get PTime. The proof of Thm. 3 uses a labelling algorithm as for Thm. 1 and PTime is guaranteed as soon as $\mathcal{M}, s \Vdash \mathsf{Kh}^{\vec{b}}_{\mathtt{a}}(\mathtt{p}, \mathtt{q})$ can be checked in PTime. Now, $\mathcal{M}, s \Vdash \mathsf{Kh}^{\vec{b}}_{\mathtt{a}}(\mathtt{p}, \mathtt{q})$ iff there is $\mathcal{A} \in \mathrm{U}_{\mathtt{a}}$ such that (1), (2) and (3) hold. There is a linear amount of automata in $\mathrm{U}_{\mathtt{a}}$ and Thm. 2 guarantees that (1) and (2) can be checked in PTime. The remaining bit is to check that (3) can be done in PTime. Notice that when (1) and (2) hold true, the first statement of Lemma 6 is equivalent to (3) being false. Thus, checking (3) follows from the combination of Lemmas 5 and 6.

5.2 Extending the Logic $\mathcal{L}_{kh}$ with Budgets

Let $\mathcal{L}_{kh}(r)$ be the ability-based logic $\mathcal{L}_{kh}$ augmented with $r \geqslant 0$ resource types. The logic $\mathcal{L}_{kh}(\star)$ denotes the version in which the number of resource types is arbitrary.

Models of $\mathcal{L}_{kh}(r)$ are of the form $\mathcal{S} = (\mathrm{S}, (\mathrm{R}_{\mathtt{a}})_{\mathtt{a} \in \mathsf{Act}}, \mathit{wf}, \mathrm{V})$ where $\mathit{wf} : \mathrm{S} \times \mathsf{Act} \rightarrow \mathbb{Z}^r$ is a weight function. The relation $\Vdash$ is updated as follows.

$\mathcal{S}, s \Vdash \mathsf{Kh}^{\vec{b}}(\varphi, \psi) \stackrel{\text{def}}{\Leftrightarrow}$ there is a plan $\sigma \in \mathsf{Act}^*$ such that (1) and (2) as for $\mathcal{L}_{kh}$, and (3) σ is $\vec{b}$-compatible at $[\![\varphi]\!]^{\mathcal{M}}$.

We already showed that the conditions (1) and (2) from $\mathcal{L}_{kh}$ can be encoded by a finite-state automaton of exponential size, and that $\mathrm{MC}(\mathcal{L}_{kh})$ is PSpace-complete. However, $\mathrm{MC}(\mathcal{L}_{kh}(\star))$ witnesses at least an exponential blow-up, as stated below, partly due to condition (3) combined with (2).

Lemma 7. $\mathrm{MC}(\mathcal{L}_{kh}(\star))$ *is* ExpSpace*-hard.*

The proof is by reduction from the control-state reachability problem for VASS, written **CREACH(VASS)**, known to be ExpSpace-complete, see e.g. (Lipton 1976; Rackoff 1978). CREACH(VASS) takes as inputs a VASS $\mathcal{V}$, a configuration $(q_0, \vec{x}_0)$, and a location q_f and asks whether there is a run from $(q_0, \vec{x}_0)$ to a configuration with location q_f.

Cor. 1 states that small witness plans in $\mathcal{L}_{kh}$ have length at most exponential in the joint size of the LTS and the formula. By contrast, since the proof of Lemma 7 uses a reduction from CREACH(VASS) for which witness runs can be of length doubly-exponential in the size of the input VASS (Lipton 1976), the witness plans in $\mathcal{L}_{kh}(\star)$ may have length doubly-exponential too (if not more). Up to now, no known upper bound exists for the length of witness plans in $\mathcal{L}_{kh}(\star)$ and the decidability status of $\mathrm{MC}(\mathcal{L}_{kh}(\star))$ is open.

However, it is possible to characterise the complexity of a natural fragment of $\mathrm{MC}(\mathcal{L}_{kh}(\star))$ by requiring a simple restriction on the weight function in LTS: we consider below the subproblem of $\mathrm{MC}(\mathcal{L}_{kh}(\star))$ in which the action costs do not depend on the states the actions are triggered. Hence, to conclude, we assume that wf is of the form $\mathsf{Act} \rightarrow \mathbb{Z}^r$.

Theorem 4. $\mathrm{MC}(\mathcal{L}_{kh}(\star))$ *restricted to LTS with action costs independent of states is* ExpSpace*-complete. For $r \in \mathbb{N}$, its restriction to r resources is* PSpace*-complete.*

ExpSpace-hardness is inherited from the proof of Lemma 7. For ExpSpace-easiness, we show that testing $\mathcal{S}, s \Vdash \mathsf{Kh}^{\vec{b}}(\mathtt{p}, \mathtt{q})$ can be reduced to an instance of CREACH(VASS) that can be checked in nondeterministic exponential space (refined analysis required here). When r is fixed, we regain PSpace.

6 Concluding Remarks

We investigated the complexity of the model-checking problem for ability-based logics, possibly with plans constrained

by budget-like requirements and/or by regularity constraints. First, we established that for the well-known knowing how logic introduced in (Wang 2015), the problem is PSpace-complete (Thm. 1). Interestingly, we have shown that the set of witness plans for a given Kh-formula is regular, a property used all along the paper. Then, we propose a generalisation of the uncertainty-based logic of (Areces et al. 2021), in which each equivalence class over sets of plans is defined by a regular language. We show that model-checking for this extension $\mathcal{L}^{\mathrm{U}}_{reg}$ is in PTime, using an algorithm based on graph accessibility (Thm. 2). The final part of the paper is devoted to add budget-like constraints to $\mathcal{L}_{kh}$ and $\mathcal{L}^{\mathrm{U}}_{reg}$, following a paradigm used in many formalisms. While model-checking for the extension $\mathcal{L}^{\mathrm{U}}_{reg}(\star)$ is shown in PTime (Thm. 3), we prove that for $\mathcal{L}_{kh}(\star)$ is ExpSpace-hard (Lemma 7). Moreover, the restriction in which the action costs do not depend on states, a reasonable assumption in many contexts, is ExpSpace-complete (Thm. 4). In all cases, we rely on results from vector addition systems.

A puzzling question remains the decidability status of $\mathrm{MC}(\mathcal{L}_{kh}(\star))$. Our investigations can be also broadened by going beyond sequential plans (see e.g. (Li and Wang 2021b)) or by investigating alternative knowing how modalities (see e.g. (Fervari et al. 2017)).

Acknowledgments

We thank the reviewers for their constructive comments and suggestions. Stéphane Demri is supported by Centre National de la Recherche Scientifique. Raul Fervari is supported by ANPCyT-PICT-2020-3780, CONICET PIP 11220200100812CO, and by the Laboratoire International Associé SINFIN.

References

Alechina, N.; Bulling, N.; Logan, B.; and Nguyen, H. 2017. The virtues of idleness: A decidable fragment of resource agent logic. *Artificial Intelligence* 245:56–85.

Alechina, N.; Bulling, N.; Demri, S.; and Logan, B. 2018. On the complexity of resource-bounded logics. *Theoretical Computer Science* 750:69–100.

Alur, R.; Henzinger, T.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

Areces, C.; Fervari, R.; Saravia, A.; and Velázquez-Quesada, F. 2021. Uncertainty-based semantics for multi-agent knowing how logics. In *TARK'21*, volume 335 of *EPTCS*, 23–37.

Blackburn, P.; de Rijke, M.; and Venema, Y. 2001. *Modal Logic*. Cambridge University Press.

Bouyer, P.; Fahrenberg, U.; Larsen, K.; Markey, N.; and Srba, J. 2008. Infinite runs in weighted timed automata with energy constraints. In *FORMATS'08*, volume 5215 of *LNCS*, 33–47. Springer.

Bulling, N., and Goranko, V. 2022. Combining quantitative and qualitative reasoning in concurrent multi-player games. *Autonomous Agents and Multi-Agent Systems* 36(2):1–33.

Cao, R., and Naumov, P. 2017. Budget-constrained dynamics in multiagent systems. In *IJCAI'17*, 915–921. ijcai.org.

Chatterjee, K.; Doyen, L.; and Henzinger, T. 2017. The cost of exactness in quantitative reachability. In *Models, Algorithms, Logics and Tools - Essays Dedicated to Kim Guldstrand Larsen on the Occasion of His 60th Birthday*, volume 10460 of *LNCS*, 367–381. Springer.

Cormen, T.; Leiserson, C.; Rivest, R. L.; and Stein, C. 2022. *Introduction to Algorithms, 4th Edition*. MIT Press.

Demri, S.; Goranko, V.; and Lange, M. 2016. *Temporal Logics in Computer Science*. Cambridge University Press.

Fervari, R.; Herzig, A.; Li, Y.; and Wang, Y. 2017. Strategically knowing how. In *IJCAI'17*, 1031–1038. ijcai.org.

Galil, Z. 1976. Hierarchies of complete problems. *Acta Informatica* 6:77–88.

Herzig, A., and Troquard, N. 2006. Knowing how to play: uniform choices in logics of agency. In *AAMAS'06*, 209–216. ACM.

Herzig, A. 2015. Logics of knowledge and action: critical analysis and challenges. *Autonomous Agents and Multi Agent Systems* 29(5):719–753.

Jamroga, W., and Ågotnes, T. 2007. Constructive knowledge: what agents can achieve under imperfect information. *Journal of Applied Non Classical Logics* 17(4):423–475.

Karp, R. M., and Miller, R. E. 1969. Parallel program schemata. *JCSS* 3(2):147–195.

Kozen, D. 1977. Lower bounds for natural proof systems. In *FOCS'77*, 254–266. IEEE Computer Society.

Li, Y., and Wang, Y. 2017. Achieving while maintaining: - A logic of knowing how with intermediate constraints. In *ICLA'17*, volume 10119 of *LNCS*, 154–167. Springer.

Li, Y., and Wang, Y. 2021a. Knowing how to plan. In *TARK'21*, volume 335 of *EPTCS*, 233–247.

Li, Y., and Wang, Y. 2021b. Planning-based knowing how: A unified approach. *Artificial Intelligence* 296:103487.

Li, Y.; Yu, Q.; and Wang, Y. 2017. More for free: a dynamic epistemic framework for conformant planning over transition systems. *JLC* 27(8):2383–2410.

Li, Y. 2017. *Knowing what to do: a logical approach to planning ad knowing how*. Ph.D. Dissertation, University of Groningen.

Lipton, R. 1976. The reachability problem requires exponential space. Technical Report 62, Department of Computer Science, Yale University.

Naumov, P., and Tao, J. 2018a. Second-order know-how strategies. In *AAMAS'18*, 390–398.

Naumov, P., and Tao, J. 2018b. Strategic coalitions with perfect recall. In *AAAI'18*, 4702–4709. AAAI Press.

Naumov, P., and Tao, J. 2018c. Together we know how to achieve: An epistemic logic of know-how. *Artificial Intelligence* 262:279–300.

Rackoff, C. 1978. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science* 6(2):223–231.

Savitch, W. 1970. Relationships between nondeterministic and deterministic tape complexities. *JCSS* 4(2):177–192.

Smith, D., and Weld, D. 1998. Conformant graphplan. In *AAAI'98*, 889–896. AAAI Press / The MIT Press.

van der Hoek, W., and Lomuscio, A. 2003. Ignore at your peril - towards a logic for ignorance. In *AAMAS'03*, 1148–1149. ACM.

van Ditmarsch, H.; Halpern, J.; van der Hoek, W.; and Kooi, B., eds. 2015. *Handbook of Epistemic Logic*. College Pub.

Wang, Y. 2015. A logic of knowing how. In *LORI'15*, volume 9394 of *LNCS*, 392–405. Springer.

Wang, Y. 2018a. Beyond knowing that: a new generation of epistemic logics. In van Ditmarsch, H., and Sandu, G., eds., *Jaakko Hintikka on knowledge and game theoretical semantics*. Springer. 499–533.

Wang, Y. 2018b. A logic of goal-directed knowing how. *Synthese* 195(10):4419–4439.