Model-Checking for Ability-Based Logics with Constrained Plans

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

[1]Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF, France

[2]FAMAF, Universidad Nacional de Córdoba / CONICET, Argentina

[3]Guangdong Technion - Israel Institute of Technology, China

37$^{\text{th}}$ AAAI Conference on Artificial Intelligence (AAAI-23) - 2023

Ability-Based Logics

Formal foundations for strategic reasoning and epistemic planning.

[van Dimarsch et al., (2015)]

A realm of logics featuring abilities:

- Propositional Dynamic Logic. [Pratt (1976), Harel (1984)]
- Knowledge and action. [Moore (1985)]
- STIT (sees to it at) logics. [Belnap & Perloff (1988)]
- Knowledge modalities and abilities. [van der Hoek & Lomuscio (2003)] [Herzig & Troquard (2006)]
- Knowing how logics. [Wang (2015)] [Areces et al. (2021)]

Our Motivations

- Knowing how $+$ numerical constraints and/or regularity constraints.
- Model checking (instead of satisfiability/validity):
  - better reflects expressivity.
- Connections with formal language theory.

A Simple Logic of Knowing How - Models

Definition (Models).

An LTS is a tuple $\mathcal{S} = (\mathsf{S}, (\mathsf{R}_a)_{a \in \mathsf{Act}}, \mathsf{V})$ where:

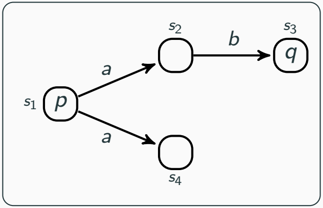
- S is a countable set of states
- $\mathsf{V} : \mathsf{S} \to 2^{\mathsf{Prop}}$
- $\mathsf{R}_a \subseteq \mathsf{S} \times \mathsf{S}$, for each $a \in \mathsf{Act}$.

A Simple Logic of Knowing How - Models

Definition (Models).

An LTS is a tuple $\mathcal{S} = (\mathsf{S}, (\mathsf{R}_a)_{a\in\mathsf{Act}}, \mathsf{V})$ where:

- S is a countable set of states
- $\mathsf{V} : \mathsf{S} \to 2^{\mathsf{Prop}}$
- $\mathsf{R}_a \subseteq \mathsf{S} \times \mathsf{S}$, for each $a \in \mathsf{Act}$.

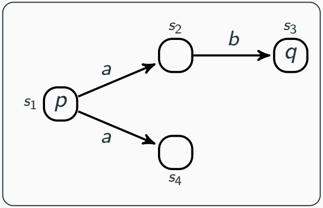


A Simple Logic of Knowing How - Models

Definition (Models).

An LTS is a tuple $\mathcal{S} = (\mathsf{S}, (\mathsf{R}_a)_{a\in\mathsf{Act}}, \mathsf{V})$ where:

- S is a countable set of states
- $\mathsf{V} : \mathsf{S} \to 2^{\mathsf{Prop}}$
- $\mathsf{R}_a \subseteq \mathsf{S} \times \mathsf{S}$, for each $a \in \mathsf{Act}$.



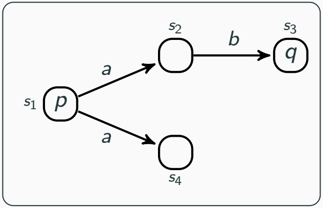
A transition a from s_1 to s_2 is read as "after executing action a at state s_1, the agent reaches state s_2".

A Simple Logic of Knowing How - Models

Definition (Models).

An LTS is a tuple $\mathcal{S} = (\mathsf{S}, (\mathsf{R}_a)_{a \in \mathsf{Act}}, \mathsf{V})$ where:

- S is a countable set of states
- $\mathsf{V} : \mathsf{S} \to 2^{\mathsf{Prop}}$
- $\mathsf{R}_a \subseteq \mathsf{S} \times \mathsf{S}$, for each $a \in \mathsf{Act}$.

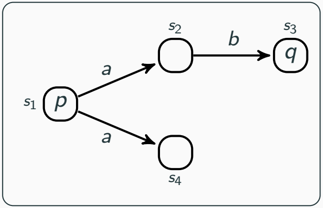


A transition a from s_1 to s_2 is read as "after executing action a at state s_1, the agent reaches state s_2".

For a set of actions Act, a plan σ is an element from Act^* (finite sequences of symbols from Act, such as a, ab and the empty plan ϵ).

Strong executability

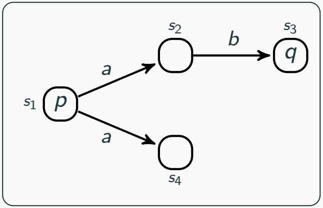
A plan must be fail-proof: each partial execution must be completed.



ab is not SE at s_1

Strong executability

A plan must be fail-proof: each partial execution must be completed.

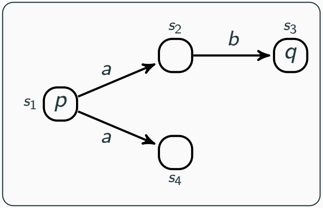


ab is not SE at s_1

We say that $(s_1, s_3) \in \mathrm{R}_{ab}$ ($s_3 \in \mathrm{R}_{ab}(s_1)$).

Strong executability

A plan must be fail-proof: each partial execution must be completed.



ab is not SE at s_1

We say that $(s_1, s_3) \in \mathrm{R}_{ab}$ $(s_3 \in \mathrm{R}_{ab}(s_1))$.

Definition (SE).

A plan $\sigma \in \mathrm{Act}^*$ 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 $\mathrm{R}_{\sigma[k+1]}(t) \neq \emptyset$.

Define the set: $\mathrm{SE}(\sigma) \stackrel{\text{def}}{=} \{s \in \mathrm{S} \mid \sigma \text{ is SE at } s\}$.

Definition (Syntax of $\mathcal{L}_{kh}$).

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

$\mathsf{Kh}(\psi, \varphi)$: *"whenever ψ holds, the agent knows how to achieve φ"*.

A Simple Logic of Knowing How - $\mathcal{L}_{kh}$

Definition (Syntax of $\mathcal{L}_{kh}$).

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

$\mathsf{Kh}(\psi, \varphi)$: *"whenever ψ holds, the agent knows how to achieve φ"*.

Definition (Semantics).

$\mathcal{S}, s \Vdash p$ iff $p \in \mathsf{V}(s)$

A Simple Logic of Knowing How - $\mathcal{L}_{kh}$

Definition (Syntax of $\mathcal{L}_{kh}$).

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

$\mathsf{Kh}(\psi, \varphi)$: *"whenever ψ holds, the agent knows how to achieve φ"*.

Definition (Semantics).

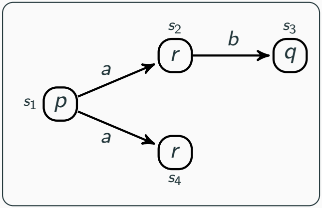
$\mathcal{S}, s \Vdash p$ iff $p \in \mathsf{V}(s)$

$\mathcal{S}, s \Vdash \mathsf{Kh}(\psi, \varphi)$ iff there exists a plan $\sigma \in \mathsf{Act}^*$ such that:

1. $[\![\psi]\!]^{\mathcal{S}} \subseteq \mathsf{SE}(\sigma)$, *and*
2. *for every* $t \in [\![\psi]\!]^{\mathcal{S}}$, $\mathsf{R}_\sigma(t) \subseteq [\![\varphi]\!]^{\mathcal{S}}$,

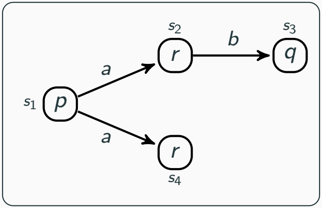
where: $[\![\chi]\!]^{\mathcal{S}} \stackrel{def}{=} \{t \mid \mathcal{S}, t \Vdash \chi\}$.

Example



$\mathcal{S}, s_1 \Vdash \mathsf{Kh}(p, r)$

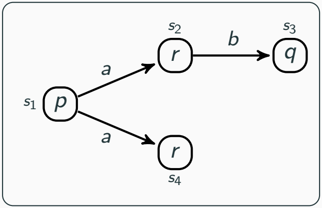
Example



$\mathcal{S}, s_1 \Vdash \mathsf{Kh}(p, r)$

the plan a is SE s_1 (the only p-state), and takes the agent from p only to r-states.

Example

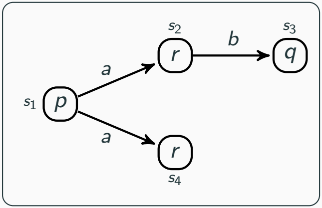


$\mathcal{S}, s_1 \Vdash \mathrm{Kh}(p, r)$

the plan a is SE s_1 (the only p-state), and takes the agent from p only to r-states.

$\mathcal{S}, s_1 \nVdash \mathrm{Kh}(p, q)$

Example



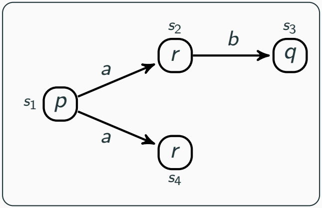
$\mathcal{S}, s_1 \Vdash \mathsf{Kh}(p, r)$

the plan a is SE s_1 (the only p-state), and takes the agent from p only to r-states.

$\mathcal{S}, s_1 \nVdash \mathsf{Kh}(p, q)$

- ϵ and a: are SE at s_1 (p-state), but do not lead to q;

Example



$\mathcal{S}, s_1 \Vdash \mathsf{Kh}(p, r)$

the plan a is SE s_1 (the only p-state), and takes the agent from p only to r-states.

$\mathcal{S}, s_1 \nVdash \mathsf{Kh}(p, q)$

- ϵ and a: are SE at s_1 (p-state), but do not lead to q;
- ab is **not** SE at s_1.

Complexity of Model Checking

Theorem

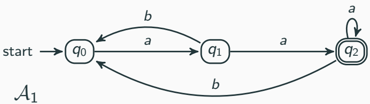
The model checking problem for $\mathcal{L}_{kh}$ is PSpace*-complete.*

Proof Strategy.

1. **Lower bound:** *reduction from non-emptiness of the intersection of Finite State Automata (*PSpace*-complete).*
2. **Upper Bound:** PSpace *algorithm, relying on a small plan property.*

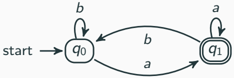
Lower bound - PSpace-**hardness**

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two automata (the argument can be extended to n automata):



$\mathcal{A}_1$

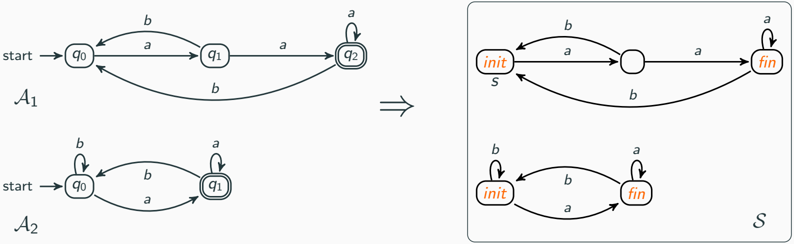
$\Rightarrow$



$\mathcal{A}_2$

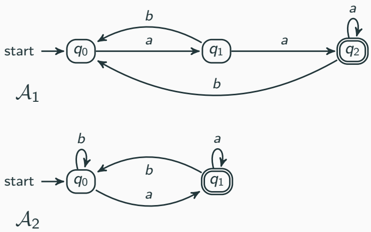
Lower bound - PSpace-hardness

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two automata (the argument can be extended to n automata):

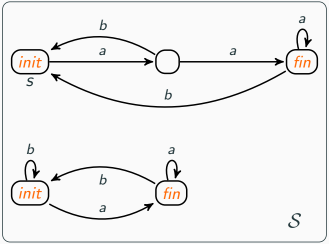


Lower bound - PSpace-hardness

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two automata (the argument can be extended to n automata):



$\Rightarrow$



Lemma.
$\mathrm{L}(\mathcal{A}_1) \cap \mathrm{L}(\mathcal{A}_2) \neq \emptyset$ if and only if $\mathcal{S}, s \Vdash \mathrm{Kh}(\mathit{init}, \mathit{fin})$.

PSpace **upper bound**

- Based on a small plan property.

PSpace **upper bound**

- Based on a small plan property. Let s a state in an LTS. Then:

Lemma.
The set $L_s = \{\sigma \mid s \in \mathsf{SE}(\sigma)\}$ is a regular language.

PSpace **upper bound**

- Based on a small plan property. Let s a state in an LTS. Then:

Lemma.
The set $L_s = \{\sigma \mid s \in \mathsf{SE}(\sigma)\}$ is a regular language.

- We intersect L_s with the set of plans witnessing a formula $\mathsf{Kh}(\psi, \varphi)$:

Lemma.
$\mathcal{S}, s \Vdash \mathsf{Kh}(\psi, \varphi)$ iff there is a plan σ of exponential size, witnessing the truth of $\mathsf{Kh}(\psi, \varphi)$.

PSpace **upper bound**

- Based on a small plan property. Let s a state in an LTS. Then:

Lemma.

The set $L_s = \{\sigma \mid s \in \mathsf{SE}(\sigma)\}$ is a regular language.

- We intersect L_s with the set of plans witnessing a formula $\mathsf{Kh}(\psi, \varphi)$:

Lemma.

$\mathcal{S}, s \Vdash \mathsf{Kh}(\psi, \varphi)$ iff there is a plan σ of exponential size, witnessing the truth of $\mathsf{Kh}(\psi, \varphi)$.

Corollary.

Checking $\mathcal{S}, s \Vdash \mathsf{Kh}(\psi, \varphi)$ can be done in polynomial space.

Constrained Plans - Regularity

Let $\mathcal{S} = (\mathsf{S}, (\mathsf{R}_a)_{a \in \mathsf{Act}}, (\mathsf{U}_{\mathfrak{a}})_{\mathfrak{a} \in \mathsf{Agt}}, \mathsf{V})$, where for every **agent** $\mathfrak{a} \in \mathsf{Agt}$, $\mathsf{U}_{\mathfrak{a}} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ (for all $\mathcal{A}_j, \mathcal{A}_k \in \mathsf{U}_{\mathfrak{a}}$, $j \neq k$ implies $\mathrm{L}(\mathcal{A}_j) \cap \mathrm{L}(\mathcal{A}_k) = \emptyset$).

Constrained Plans - Regularity

Let $\mathcal{S} = (\mathsf{S}, (\mathsf{R}_a)_{a \in \mathsf{Act}}, (\mathsf{U}_{\mathfrak{a}})_{\mathfrak{a} \in \mathsf{Agt}}, \mathsf{V})$, where for every **agent** $\mathfrak{a} \in \mathsf{Agt}$, $\mathsf{U}_{\mathfrak{a}} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ (for all $\mathcal{A}_j, \mathcal{A}_k \in \mathsf{U}_{\mathfrak{a}}$, $j \neq k$ implies $\mathrm{L}(\mathcal{A}_j) \cap \mathrm{L}(\mathcal{A}_k) = \emptyset$).

Definition (Semantics of $\mathcal{L}^{\mathrm{U}}_{reg}$).

$\mathcal{S}, s \Vdash \mathsf{Kh}_{\mathfrak{a}}(\psi, \varphi)$ iff there exists $\mathcal{A} \in \mathsf{U}_{\mathfrak{a}}$ such that, **for every** $\sigma \in \mathrm{L}(\mathcal{A})$:

1. $[\![\psi]\!]^{\mathcal{S}} \subseteq \mathsf{SE}(\sigma)$, *and*
2. *for every* $t \in [\![\psi]\!]^{\mathcal{S}}$, $\mathsf{R}_\sigma(t) \subseteq [\![\varphi]\!]^{\mathcal{S}}$ *(* $[\![\chi]\!]^{\mathcal{S}} \stackrel{\mathit{def}}{=} \{t \mid \mathcal{S}, t \Vdash \chi\}$*).*

Constrained Plans - Regularity

Let $\mathcal{S} = (\mathsf{S}, (\mathsf{R}_a)_{a \in \mathsf{Act}}, (\mathsf{U}_{\mathfrak{a}})_{\mathfrak{a} \in \mathsf{Agt}}, \mathsf{V})$, where for every **agent** $\mathfrak{a} \in \mathsf{Agt}$, $\mathsf{U}_{\mathfrak{a}} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ (for all $\mathcal{A}_j, \mathcal{A}_k \in \mathsf{U}_{\mathfrak{a}}$, $j \neq k$ implies $\mathrm{L}(\mathcal{A}_j) \cap \mathrm{L}(\mathcal{A}_k) = \emptyset$).

Definition (Semantics of $\mathcal{L}^{\mathrm{U}}_{reg}$).

$\mathcal{S}, s \Vdash \mathsf{Kh}_{\mathfrak{a}}(\psi, \varphi)$ iff there exists $\mathcal{A} \in \mathsf{U}_{\mathfrak{a}}$ such that, **for every** $\sigma \in \mathrm{L}(\mathcal{A})$:

1. $[\![\psi]\!]^{\mathcal{S}} \subseteq \mathsf{SE}(\sigma)$, *and*
2. *for every* $t \in [\![\psi]\!]^{\mathcal{S}}$, $\mathsf{R}_\sigma(t) \subseteq [\![\varphi]\!]^{\mathcal{S}}$ *(* $[\![\chi]\!]^{\mathcal{S}} \overset{def}{=} \{t \mid \mathcal{S}, t \Vdash \chi\}$*).*

Theorem

Model checking $\mathcal{L}^{\mathrm{U}}_{reg}$ is in PTime.

Constrained Plans - Regularity

Let $\mathcal{S} = (\mathsf{S}, (\mathsf{R}_a)_{a \in \mathsf{Act}}, (\mathsf{U}_{\mathfrak{a}})_{\mathfrak{a} \in \mathsf{Agt}}, \mathsf{V})$, where for every **agent** $\mathfrak{a} \in \mathsf{Agt}$, $\mathsf{U}_{\mathfrak{a}} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots\}$ (for all $\mathcal{A}_j, \mathcal{A}_k \in \mathsf{U}_{\mathfrak{a}}$, $j \neq k$ implies $\mathrm{L}(\mathcal{A}_j) \cap \mathrm{L}(\mathcal{A}_k) = \emptyset$).

Definition (Semantics of $\mathcal{L}^{\mathrm{U}}_{reg}$).

$\mathcal{S}, s \Vdash \mathsf{Kh}_{\mathfrak{a}}(\psi, \varphi)$ iff there exists $\mathcal{A} \in \mathsf{U}_{\mathfrak{a}}$ such that, **for every** $\sigma \in \mathrm{L}(\mathcal{A})$:

1. $[\![\psi]\!]^{\mathcal{S}} \subseteq \mathsf{SE}(\sigma)$, *and*
2. *for every* $t \in [\![\psi]\!]^{\mathcal{S}}$, $\mathsf{R}_\sigma(t) \subseteq [\![\varphi]\!]^{\mathcal{S}}$ *(* $[\![\chi]\!]^{\mathcal{S}} \stackrel{def}{=} \{t \mid \mathcal{S}, t \Vdash \chi\}$*).*

Theorem

Model checking $\mathcal{L}^{\mathrm{U}}_{reg}$ is in PTime.

Proof strategy: *Algorithm based on reachability checks of a product graph ($\mathcal{S} \times \mathcal{A}$).*

Adding budgets

- In many situations, actions have costs (and executions must stay within a certain budget).
- Consider a function $\mathit{wf} : \mathsf{S} \times \mathsf{Act} \to \mathbb{Z}^r$, for some $r \geq 0$ (a number of resources).
- $\mathsf{Kh}^{\vec{b}}(\psi, \varphi)$: *knowing how to make φ true, given ψ, with budget $\vec{b}$.*

Adding budgets

- In many situations, actions have costs (and executions must stay within a certain budget).
- Consider a function $\mathit{wf} : \mathsf{S} \times \mathsf{Act} \to \mathbb{Z}^r$, for some $r \geq 0$ (a number of resources).
- $\mathsf{Kh}^{\vec{b}}(\psi, \varphi)$: *knowing how to make φ true, given ψ, with budget $\vec{b}$.*

Theorem

1. Model-checking $\mathcal{L}^{\mathrm{U}}_{reg}$ + budgets: PTime.

   Proof strategy: Non-safety problem in Vector Addition Systems (VASS).

Adding budgets

- In many situations, actions have costs (and executions must stay within a certain budget).
- Consider a function $\mathit{wf} : \mathsf{S} \times \mathsf{Act} \to \mathbb{Z}^r$, for some $r \geq 0$ (a number of resources).
- $\mathsf{Kh}^{\vec{b}}(\psi, \varphi)$: *knowing how to make φ true, given ψ, with budget $\vec{b}$.*

Theorem

1. Model-checking $\mathcal{L}_{reg}^{\mathrm{U}}$ + budgets: PTime.

   Proof strategy: Non-safety problem in Vector Addition Systems (VASS).

2. Model-checking $\mathcal{L}_{kh}$ + budgets: ExpSpace-hard (no upper-bound).

   Proof strategy: Reduction from the control-state reachability problem for VASS.

Final remarks

- We studied complexity of model-checking for ability-based logics.
  - Linear plans ([Wang (2015)]).
  - Regularity constraints (ext. [Areces et al. (2021)]).
  - Budget constraints.
  - Results for variant logics.

Final remarks

- We studied complexity of model-checking for ability-based logics.
  - Linear plans ([Wang (2015)]).
  - Regularity constraints (ext. [Areces et al. (2021)]).
  - Budget constraints.
  - Results for variant logics.
- Future work:
  - Exact complexity of $\mathcal{L}_{kh}$ + budgets.
  - Other constraints (e.g. non linear plans).
  - Other semantics for knowing how (e.g. [Fervari et al. (2017)]).