Artificial Intelligence 296 (2021) 103487

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Planning-based knowing how: A unified approach ☆

Yanjun Li [a], Yanjing Wang [b,c,*]

[a] College of Philosophy, Nankai University, China
[b] Center for Philosophy and the Future of Humanity, Peking University, China
[c] Department of Philosophy, Peking University, China

ARTICLE INFO

Article history:
Received 2 March 2019
Accepted 17 February 2021
Available online 24 February 2021

Keywords:
Epistemic planning
Know-how
Epistemic logic
Conformant planning
Contingent planning
Knowledge-based programs

ABSTRACT

Various logical notions of know-how have been recently proposed and studied in the literature based on different types of epistemic planning in different frameworks. This paper proposes a unified logical framework to incorporate the existing and some new notions of know-how. We define the semantics of the know-how operator using a unified notion of epistemic planning with parameters of different types of plans specified by a programming language. Surprisingly, via a highly unified completeness proof, we show that all the ten intuitive notions of plans discussed in this paper lead to exactly the same know-how logic, which is proven to be decidable. We also show that over finite models, the know-how logic based on knowledge-based plans requires an extension with an axiom capturing the compositionality of the plans.

In the context of epistemic planning, our axiomatization results reveal the core principles behind the very idea of epistemic planning, independent of the particular notion of plans. Moreover, since epistemic planning can be expressed by the know-how modality in our object language, we can greatly generalize the planning problems that can be solved formally by model checking various formulas in our language.

1. Introduction

Automated planning in AI studies how to find executable plans to achieve certain goals. Technically, the classical planning amounts to the reachability problem in a transition system, usually generated from compact representations of possible states and actions. Things become more exciting and more realistic when uncertainty kicks in, which may come from different sources, such as uncertainty of the initial state, the observational power of the agent, and the non-deterministic actions (cf. [19,18]). Based on the presence of different types of uncertainty, we can classify various (non-probabilistic) planning problems accordingly as follows:

☆ This article belongs to Special Issue: Epistemic Planning.

* Corresponding author at: Department of Philosophy, Peking University, China.
E-mail addresses: lyjlogic@nankai.edu.cn (Y. Li), y.wang@pku.edu.cn (Y. Wang).

https://doi.org/10.1016/j.artint.2021.103487
0004-3702/

Types of Planning \ Uncertainty	Init	Obs	Act
Classical	no	full	no
Fully Observable Non-deterministic (FOND)	no	full	yes
Conformant	yes	none	yes
Contingent	yes	partial	yes

For example, *contingent planning* is the planning problem where there can be some uncertainty about the initial state ("yes" in the column **Init**), and the planner cannot always fully observe the current state during the execution of the plan (hence "partial" in the column **Obs**), and the actions may have non-deterministic effects ("yes" in **Act**). The categorization can be further refined by considering whether there are multiple agents, whether the actions can be executed concurrently, and so on (cf. e.g., [19]).

A natural way to formally handle uncertainty qualitatively is to use the notions of *knowledge and belief*, as studied in epistemic logic. Recent years witnessed a growing interest in automated planning using epistemic logic, which is particularly useful in the following planning scenarios where

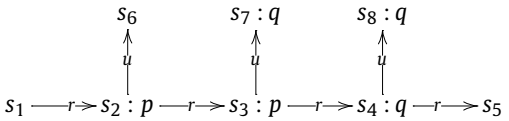
- the goal of planning is epistemic, e.g., the agent knows that φ;
- actions change epistemic states of the agents, e.g., public announcements;
- multiple agents are involved;
- agents need to act with respect to their own knowledge and belief.

In particular, dynamic epistemic logic (DEL) plays an important role in handling the uncertainty progression and regression in the reasoning behind planning [10,33,3,5,60,42,21,11,36]. However, the multi-agent automated planning problem using the standard DEL framework is usually undecidable [10,6]. For the single-agent case, things are more promising [10,33,60,4]. For example, we can formalize the (generalized) conformant planning as a model-checking problem in some dynamified version of epistemic PDL [59,32].

At the same time, the logical formalization of various planning problems also gives rise to a new type of epistemic logic, i.e., the logic of *knowing how*. Knowledge-how is attracting more and more attention in epistemology nowadays (cf. e.g., [46,45,15]), and has been discussed in the early literature of AI [35,34]. In [49,51], Wang proposed a new way of formalizing knowing how via the idea of planning. Intuitively, an agent knows how to achieve φ if there is a plan such that the agent knows that it is a good plan to achieve φ according to some formal notion of planning. This approach is inspired by the general research program on epistemic logic of *knowing-wh* proposed by Wang in [52], such as knowing whether [14], knowing why [58], knowing what [55], and so on. It is crucial to note that the semantics of the know-how operator is roughly in the shape of $\exists\sigma\mathcal{K}(\sigma \ \texttt{guarantees} \ \varphi)$, where the order of the quantifier and the modality $\mathcal{K}$ is important, which reflects the distinction of *de re* and *de dicto* in philosophy.[1] This general notion of know-how fits very well with the notion of planning under uncertainty, such as the conformant and contingent planning mentioned above.

The idea of conformant planning (cf. [44]) is to find a finite linear plan which is always executable to the end to make sure φ, given the uncertainty of both the initial state and the effects of actions. As an example, consider the following scenario taken from [56].

Example 1. You are a rookie spy sneaking in an enemy's building with a map of the floor at hand illustrated below, where rooms (denoted by s_i) are connected by (one-way) corridors (r arrows) or stairs (u arrows). Some enemy spotted you, and you have lost your way in a panic such that you are not sure in which p room (s_2 or s_3) you actually are. You have to move quickly to some safe places (the q rooms s_4, s_7, and s_8). Now, do you know how to be safe given the uncertainty of where you are?



Although the agent is not sure which p room is the current location, he still *knows how* to reach q given p, because there is a conformant plan ru that makes sure the agent will reach one of the q rooms from *any* p room. Intuitively, the p rooms amount to the initial uncertainty for an implicit $\mathcal{K}$ operator. Formally, *knowing how* to achieve q given p is expressed by a binary *global* modality $\mathcal{K}h(p, q)$ in [49].[2] In general, we may use the transition systems over some state space to model the agent's ability to act, and ask whether the agent knows how to achieve some propositional goals.

[1] See the critical survey by Herzig [20].

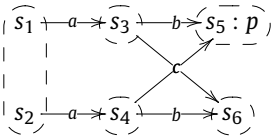
[2] Here *global* means that the truth value of the $\mathcal{K}h$ formula does not depend on the current state. See [25] for the philosophical motivation.

In [30] and [28], different versions of linear-plan based on knowing how were proposed and studied. For example, in the above scenario, it seems also intuitive to say that the agent also knows how to make sure $\neg q$ given p, since he can just head right (r) until the end, while there is no single uniform plan which can always be executed successfully to the end from each of the p states. Li introduced the *weak linear plan* where the executability of the conformant planning is weakened to facilitate this [28]. Under such a notion, *rrr* is a good plan to achieve $\neg q$ given p, since you can make sure that whenever you cannot move forward you will achieve the goal.

However, there are other intuitive scenarios that cannot be handled so far. For example, an agent can be said to know how to open the safe if he knows that the password consists of finitely many 1s, and when the password is correct the door will automatically open. The plan is to simply input 1111... until the safe is opened.[3] However, such an unbounded plan is not captured by the existing work on the logic of know-how, although in AI planning, there have been interesting discussions such as [27,8].

On the other hand, linear plans are quite restrictive, in [16] a natural notion of knowing how is proposed based on *contingent planning*, where the plan can be branching based on the information on each state. In particular, the agent can act differently based on what they become to know during the execution of the plan. For example, consider the following example.

Example 2. You are not sure about the exact cause of some symptom, thus you have some uncertainty about your current state represented by the bubble over s_1 and s_2 below. After taking the test a, you will know the cause thus the current state. Taking the treatments b or c may have different effects depending on your actual state: it may cure your symptom (p) or not ($\neg p$). Now given the initial uncertainty about s_1 and s_2, do you know how to make sure p?



There is no sensible uniform *linear plan* in this scenario to reach p given the initial uncertainty $\{s_1, s_2\}$, but it is clearly reasonable to say that the agent knows how to achieve p, if conditional plans are allowed: he can first do a and then depending on his knowledge about the resulting state (s_3 or s_4) he can choose b or c respectively to make sure p. Formally we can use a unary *local modality* to express that the agent knows how to achieve p by simply $\mathcal{K}hp$ as in [16], where the usual know-that modality is also included in the language to capture the interaction of the two. Moreover, inspired by the earlier work on coalition logic and alternating temporal logic (cf. e.g., [43,2,47,1]), a notion of multi-agent distributed know-how based on *single-step* joint plans was proposed and discussed in [37,38], where all the joint actions are assumed to be always executable.

In the existing work on such logics of knowing how, the plans or strategies are basically (partial) functions from the epistemic equivalence classes (the bubbles in the above example) to actions. Essentially those plans or strategies are memory-less in the sense that the agent will do the same whenever it reaches the same knowledge state. However, as we see in the linear case, some extra information can be provided by memorizing the plan itself, e.g., first do a, then if you know p do b, otherwise do c; next, if you still know that p is the case then do a again, otherwise stop. Intuitively such a notion of plan can be independent of the concrete model in concern. To handle such plans formally, we need a *language* to specify them. Actually, the lesson on the need for a specification language was learned in the related field of knowledge-based programs [13]. Indeed, some pioneering work has been done in bringing the knowledge-based program and epistemic planning together [22,23], where the planning problem is reduced to finding a knowledge-based program which is more succinct than a standard program without epistemic conditions. The idea of using programs to specify plans is much older, which dates back to the earlier work in AI planning (cf. e.g., [26]).

Inspired by the knowledge-based-program approach to epistemic planning [22,26], in this paper, we try to unify all the existing single-agent know-how notions and some new ones by using various sublanguages of a programming language. Depending on the shape of the plan specified by the language, in principle, there can be different logics of know-how. However, is it really the case? Is there a *single logic* to rule them all? This is indeed the starting point of this paper.

Let us summarize the important questions which were left open in the existing literature of the single-agent know-how logic:

- What are the local versions[4] of the linear know-how operators and their logics in presence of the know-that operator?

[3] Compare this with the scenario where the agent only knows that the password is a finite number but has only one chance to input the password.

[4] Here "local" means the semantics of the know-how modality depends on the current state and the uncertainty, in contrast to the global know-how modality in [50].

- How to handle other intuitive notions of know-how? E.g., the ones based on unbounded linear plans or memory-sensitive conditional plans.
- Do different planning-based notions of know-how lead to the same logic?

In this paper, we answer the above questions by introducing a general framework that can unify different types of the logics of know-how.[5] Our concrete contributions in the context of the logic of knowing how are summarized as the following:

- Based on 10 types of plans, we introduce a spectrum of semantics for the know-how operator sharing a generic form, and demonstrate their differences on models.
- We show that the logic with both know-that and know-how modalities stays the same under all the 10 different semantics by proving a generic completeness result.
- When restricted to finite models, the know-how logic based on the knowledge-based plans requires one more axiom reflecting the compositionality of conditional plans.
- The two logics obtained in this paper are *decidable non-normal* modal logics.

In the context of epistemic planning, our contributions can be summarized below, to which we will come back in Section 6.

- **A specification language of epistemic plans** We use a programming language to syntactically specify various types of plans (even with unbounded loops), and we give a rigorous structural operational semantics. This allows us to talk and compare plans without referring to a particular model in concern (see Section 6.2).
- **Generalized epistemic planning as model checking** By having the know-how operator *in* the logical language, we can treat highly non-trivial planning problems by model checking formulas with negations and nested modalities such as $\mathcal{K}h(\varphi \wedge \neg\mathcal{K}h\neg\varphi)$. The generalization sometimes can be done without paying the price in computational complexity of model checking (see Section 6.6).
- **Model theory of (epistemic) planning** We compare the *expressive power* of different notions of plans in terms of the $\mathcal{K}h$ formulas based on different semantics, and ask when two different models can be viewed the same given a certain notion of planning. This opens up a way to bring (modal) model theoretical tools, such as bisimulation, to planning (see Section 6.7).
- **Proof theory of epistemic planning** Beyond the philosophical importance, our unified axiomatization reveals the core ideas of epistemic planning, independent of the particular type of plans in use. It also provides a syntactic method to reason about *high-level* planning (see Section 6.7).

The structure of the paper is as follows. In Section 2, we lay out the basics of an epistemic language and a programming language to be used to define various types of plans. Section 3 introduces various semantics of know-how based on different types of plans specified by the sublanguages of the programming language. In Section 4, we prove a uniform completeness result for all the semantics of know-how discussed in this paper. Section 5 discusses the logic over finite models. We discuss various design choices in our paper and the relevance of our work to epistemic planning in Section 6. Finally, we conclude in Section 7 with further directions.

2. Preliminaries

To discuss all the later notions of know-how in a unified way, we will introduce a programming language to specify various types of plans. Some plans may involve branching structures by testing certain conditions. To specify those conditions precisely, we first introduce a simple logical language ($\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ below) and its semantics (cf. e.g., [56]). Note that $\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ is only used to specify the conditions in the plans, and it is *not* the language of the logic of know-how, which will be introduced in the next section.

2.1. A language for the branching conditions

Definition 3 *(EAL Language).* Let $\mathbf{P}$ be a countable set of variables, and $\mathbf{A}$ be a countable set of atomic actions. The language $\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ of Epistemic Action Logic (EAL) is defined as follows:

$$\varphi \quad ::= \quad \top \mid p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \mathcal{K}\varphi \mid [a]\varphi$$

where $p \in \mathbf{P}$ and $a \in \mathbf{A}$.

[5] For convenience of presentation, we focus on the single-agent case in the technical development. However, the main results can be generalized in principle to the multi-agent case (see Section 6.3).

The auxiliary connectives and modalities $\rightarrow, \vee, \langle a \rangle$ are defined as abbreviations as usual.

The semantics of EAL formulas is defined on models with both epistemic relations and transitions labeled by atomic actions. Such models will also be used for the later know-how logics (see Section 6.5 for the choice of using such models).

Definition 4 *(Model).* An Epistemic Transition System (i.e., a model) $\mathcal{M}$ is a tuple

$$\langle W, \mathbf{A}, \sim, \{Q(a) \mid a \in \mathbf{A}\}, V \rangle$$

where:

- W is a non-empty set;
- $\mathbf{A}$ is a set of actions;
- $\sim \subseteq W \times W$ is an equivalence relation over W;
- $Q(a) \subseteq W \times W$ is a binary relation for each $a \in \mathbf{A}$;
- $V : W \rightarrow 2^{\mathbf{P}}$ is a valuation.

A frame is a model without the valuation function.

Notations We use $[s]$ to denote the set $\{t \in W \mid s \sim t\}$. The binary relation $Q(a)$ also can be seen as a function from W to the power set of W, such that for each $s \in W$, $Q(a)(s) = \{t \in W \mid (s,t) \in Q(a)\}$. Thus, by abusing the notation, we sometimes also write $(s,t) \in Q(a)$ as $t \in Q(a)(s)$. Intuitively, a model specifies the agent's abilities to act and the uncertainty about the states. Note that in contrast to [22,38], we do not assume that all the actions are executable on all the states.[6]

$\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ formulas are given truth values on pointed epistemic transition systems.

Definition 5 *(Semantics).* Given a formula $\varphi \in \mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ and a pointed model $(\mathcal{M}, s)$ where $\mathcal{M} = \langle W, \mathbf{A}, \sim, \{Q(a) \mid a \in \mathbf{A}\}, V \rangle$, the satisfaction relation on φ and pointed model $(\mathcal{M}, s)$ is defined as follows:

$\mathcal{M}, s \vDash \top$	$\Leftrightarrow$	*always*
$\mathcal{M}, s \vDash p$	$\Leftrightarrow$	$s \in \mathcal{V}(p)$
$\mathcal{M}, s \vDash \neg\varphi$	$\Leftrightarrow$	$\mathcal{M}, s \nvDash \varphi$
$\mathcal{M}, s \vDash (\varphi \wedge \psi)$	$\Leftrightarrow$	$\mathcal{M}, s \vDash \varphi$ and $\mathcal{M}, s \vDash \psi$
$\mathcal{M}, s \vDash \mathcal{K}\varphi$	$\Leftrightarrow$	for all t, if $s \sim t$ then $\mathcal{M}, t \vDash \varphi$
$\mathcal{M}, s \vDash [a]\varphi$	$\Leftrightarrow$	for all t, if $(s,t) \in Q(a)$ then $\mathcal{M}, t \vDash \varphi$

A formula is valid on a frame if it is true on all the pointed models based on that frame.

Here are some standard results about bisimulation that will be useful in the later technical discussion.

Definition 6 *(Bisimulation).* Given a model $\mathcal{M} = \langle W, \mathbf{A}, \sim, \{Q(a) \mid a \in \mathbf{A}\}, V \rangle$, a non-empty *symmetric* binary relation Z on the domain of $\mathcal{M}$ is called a bisimulation iff whenever $(w, v) \in Z$ the following conditions are satisfied:

(1). For any $p \in \mathbf{P} : p \in V(w) \iff p \in V(v)$.
(2). if $w \sim w'$ for some $w' \in W$ then there exists a $v' \in W$ such that $v \sim v'$ and $w' Z v'$.
(3). for any $a \in \mathbf{A}$, if $(w, w') \in Q(a)$ for some $w' \in W$ then there exists a $v' \in W$ such that $(v, v') \in Q(a)$ and $w' Z v'$.

The pointed models $(\mathcal{M}, w)$ and $(\mathcal{M}, v)$ are said to be *bisimilar* $(\mathcal{M}, w \leftrightarrows \mathcal{M}, v)$ if there is a bisimulation Z such that $(w, v) \in Z$.

We use $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [s']$ to denote that for each $t' \in [s']$, there exists $t \in [s]$ such that $\mathcal{M}, t \leftrightarrows \mathcal{M}, t'$ and vice versa. Given two pointed models $(\mathcal{M}, s)$ and $(\mathcal{M}, s')$, we use $\mathcal{M}, s \equiv_{\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}} \mathcal{M}, s'$ to denote that for each $\varphi \in \mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$, $\mathcal{M}, s \vDash \varphi$ if and only if $\mathcal{M}, s' \vDash \varphi$. Similarly, $\mathcal{M}, s \equiv_{\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}|_{\mathcal{K}}} \mathcal{M}, s'$ denotes that for each $\mathcal{K}\varphi \in \mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$, $\mathcal{M}, s \vDash \mathcal{K}\varphi$ if and only if $\mathcal{M}, s' \vDash \mathcal{K}\varphi$.

Please note that $\mathcal{M}, s \leftrightarrows \mathcal{M}, s'$ implies $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [s']$, but the other way around does not work in general.

As an adaption of the Hennessy-Milner theorem for modal logic (cf. e.g., [9]), we have:

Proposition 7. *Given a finite model $\mathcal{M}$, we have that $\mathcal{M}, s \equiv_{\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}} \mathcal{M}, s'$ iff $\mathcal{M}, s \leftrightarrows \mathcal{M}, s'$.*

[6] Intuitively, not all the actions are executable under all the scenarios. Moreover, even as a technical assumption, assuming universal executability may make the unbounded linear plan never terminates, e.g., recall the plan 111…for opening a safe mentioned in the introduction.

Moreover, if we only look at the $\mathcal{K}\varphi$ formulas, we have:

Proposition 8. *Given a finite model $\mathcal{M}$, $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [s']$ iff $\mathcal{M}, s \equiv_{\mathcal{L}_{\mathbf{A}}^{EAL}|_{\mathcal{K}}} \mathcal{M}, s'$.*

Proof. Left to Right: Assuming that $\mathcal{M}, s \not\equiv_{\mathcal{L}_{\mathbf{A}}^{EAL}|_{\mathcal{K}}} \mathcal{M}, s'$, it follows that there is $\mathcal{K}\varphi \in \mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ such that $\mathcal{M}, s \vDash \mathcal{K}\varphi$ but $\mathcal{M}, s' \nvDash \mathcal{K}\varphi$. It follows by $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [s']$ that there is some u such that $s \sim u$ and $\mathcal{M}, u \leftrightarrows \mathcal{M}, s'$. By $s \sim u$, we have that $\mathcal{M}, u \vDash \mathcal{K}\varphi$. By $\mathcal{M}, u \leftrightarrows \mathcal{M}, s'$ and Proposition 7, we know that $\mathcal{M}, s' \vDash \mathcal{K}\varphi$. This is contradictory to $\mathcal{M}, s' \nvDash \mathcal{K}\varphi$. Thus, if $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [s']$ then $\mathcal{M}, s \equiv_{\mathcal{L}_{\mathbf{A}}^{EAL}|_{\mathcal{K}}} \mathcal{M}, s'$.

Right to Left: Firstly, we will show that for each $t \in [s']$ there is some $u \in [s]$ such that $\mathcal{M}, u \leftrightarrows \mathcal{M}, t$. If not, it follows that there is some $t \in [s']$ such that $\mathcal{M}, t \not\leftrightarrows \mathcal{M}, u$ for each $u \in [s]$. By Proposition 7, we have that for each $u \in [s]$, there is some formula φ_u such that $\mathcal{M}, t \nvDash \varphi_u$ but $\mathcal{M}, u \vDash \varphi_u$. Since $\mathcal{M}$ is finite, the disjunction of all these φ_u is a formula, denoted ψ. From this follows that $\mathcal{M}, s \vDash \mathcal{K}\psi$ and $\mathcal{M}, t \vDash \neg\psi$. Since $t \sim s'$, $\mathcal{M}, s' \vDash \neg\mathcal{K}\psi$. This is contradictory to $\mathcal{M}, s \equiv_{\mathcal{L}_{\mathbf{A}}^{EAL}|_{\mathcal{K}}} \mathcal{M}, s'$. Thus, we have shown that for each $t \in [s']$ there is some $u \in [s]$ such that $\mathcal{M}, u \leftrightarrows \mathcal{M}, t$. By a similar process, we can show that for each $u \in [s]$ there is some $t \in [s']$ such that $\mathcal{M}, u \leftrightarrows \mathcal{M}, t$. Therefore, $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [s']$. □

2.2. A programming language for plan specification

Inspired by [22], we use a programming language with while-loops as the specification language of plans. To stay general, we do not restrict ourselves to knowledge-based programs here.

Definition 9 *(Programming language* `Prg`*).* Given a set $\mathbf{A}$ of primitive actions, a program is defined inductively as follows:

$$\pi \quad ::= \quad \epsilon \mid a \mid (\pi;\pi) \mid \texttt{if}\ \varphi\ \texttt{then}\ \pi\ \texttt{else}\ \pi \mid \texttt{while}\ \varphi\ \texttt{do}\ \pi$$

where $a \in \mathbf{A}$, $\varphi \in \mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$.

We use π^n to denote the program $\underbrace{\pi;\cdots;\pi}_{n}$ and π^0 is the empty plan ϵ. With the while-loop we can express "*keep doing a indefinitely until it is not possible, then continue with b*" by $(\texttt{while}\ \langle a\rangle\top\ \texttt{do}\ a);b$. We can also use nested loops. For example, you may have an unbounded plan to go out of a grand maze: always turn left (l) if you cannot go forward (f) anymore, until you know you are out of the maze, which can be expressed by

$$\texttt{while}\ \neg\mathcal{K}\,out\ \texttt{do}\ ((\texttt{while}\ \langle f\rangle\top\ \texttt{do}\ f);l).$$

Given a program, we need to define the traces it produces when executing it in a model. Before that, we need to define the semantics of our programs formally.

Inspired by the ideas in [41] for programming languages and process algebra, we first give a *structural operational semantics* to the programs. Note that compared to [41], we also add the label information in the transition rules.

Definition 10 *(Structural operational semantics (SOS) for* `Prg`*).* Given a model $\mathcal{M}$, the semantics of programs with respect to $\mathcal{M}$ is defined by the following rules about the transition relation in the shape of $\langle \pi, s\rangle \overset{z}{\Rightarrow} \gamma$, where $\pi \in \texttt{Prg}$, $s \in S_{\mathcal{M}}$, $z \in \mathbf{A} \cup \{\epsilon\}$, and γ is either in the form of $\langle \pi', s'\rangle$ or s', where $\pi' \in \texttt{Prg}$ and $s' \in S_{\mathcal{M}}$. We also call the rules without the premises *axioms*.

$$\textsf{Atom}\ \frac{}{\langle a, s\rangle \overset{a}{\Rightarrow} t}\ (t \in Q(s)(a)) \qquad \textsf{Skip}\ \frac{}{\langle \epsilon, s\rangle \overset{\epsilon}{\Rightarrow} s}$$

$$\textsf{CompoL}\ \frac{\langle \pi_1, s\rangle \overset{z}{\Rightarrow} \langle \pi_1', s'\rangle}{\langle \pi_1;\pi_2, s\rangle \overset{z}{\Rightarrow} \langle \pi_1';\pi_2, s'\rangle} \qquad \textsf{CompoR}\ \frac{\langle \pi_1, s\rangle \overset{z}{\Rightarrow} s'}{\langle \pi_1;\pi_2, s\rangle \overset{z}{\Rightarrow} \langle \pi_2, s'\rangle}$$

$$\textsf{IfthenP}\ \frac{}{\langle \texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2, s\rangle \overset{\epsilon}{\Rightarrow} \langle \pi_1, s\rangle}\ (\text{if } \mathcal{M}, s \vDash \varphi)$$

$$\textsf{IfthenN}\ \frac{}{\langle \texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2, s\rangle \overset{\epsilon}{\Rightarrow} \langle \pi_2, s\rangle}\ (\text{if } \mathcal{M}, s \nvDash \varphi)$$

$$\textsf{While}\ \frac{}{\langle \texttt{while}\ \varphi\ \texttt{do}\ \pi, s\rangle \overset{\epsilon}{\Rightarrow} \langle \texttt{if}\ \varphi\ \texttt{then}\ (\pi;\texttt{while}\ \varphi\ \texttt{do}\ \pi)\ \texttt{else}\ \epsilon, s\rangle}$$

We say γ is *terminal* if $\gamma = t$ for some state t, we say $\gamma = \langle \pi, s \rangle$ is *stuck* if there is no γ' and z such that $\gamma \overset{z}{\Rightarrow} \gamma'$ according to the SOS with respect to the model.

Intuitively, the expression $\langle \pi, s \rangle$ says that the program π is to be executed at s. The sequence $\langle \pi, s \rangle \overset{z}{\Rightarrow} \langle \pi', s' \rangle$ captures the transition by executing the first step z of π at s, after which we are to execute the remaining π' at s'. $\langle \pi, s \rangle \overset{z}{\Rightarrow} s'$ intuitively says after the execution of z the program π successfully terminates at s'. It is clear that $\overset{\epsilon}{\Rightarrow}$ does not change the state. Intuitively, a rule in the SOS tells us how to execute the more complicated program (in the conclusion) given the execution of some subprogram (in the premises).[7] For example, CompoR says if executing π_1 on s terminates by doing z, then executing $\pi_1; \pi_2$ on s can bring us to a state s' by z, where π_2 is to be executed.

Given a model, we can view the SOS with respect to the model as a *proof system* (with axioms and rules with a single premise) that derives "sequents" in the form of $\gamma \overset{z}{\Rightarrow} \beta$. For example, if $t \in Q(s)(a)$ in the model $\mathcal{M}$, then we have the following "proof":

$$\dfrac{\dfrac{\dfrac{}{\langle a, s \rangle \overset{a}{\Rightarrow} t}\,\text{Atom}}{\langle (a; b), s \rangle \overset{a}{\Rightarrow} \langle b, t \rangle}\,\text{CompoR}}{\langle (a; b); c, s \rangle \overset{a}{\Rightarrow} \langle b; c, t \rangle}\,\text{CompoL}$$

However, in the setting of SOS, we care more about the *sequences* consisting of such single step proofs. The SOS with respect to a model and a program can generate the *executions* of the program which we call the *derivations*, following [41].

Definition 11 *(Derivations).* Given π and a pointed model $\mathcal{M}, s$, let $\gamma_0 = \langle \pi_0, s_0 \rangle$, a (complete) *derivation* of π at s according to the SOS is either finite:

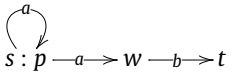
$$\gamma_0 \overset{z_0}{\Rightarrow} \gamma_1 \overset{z_1}{\Rightarrow} \cdots \overset{z_{k-1}}{\Rightarrow} \gamma_k \quad \text{where } \gamma_k \text{ is terminal or stuck}$$

or infinite:

$$\gamma_0 \overset{z_0}{\Rightarrow} \gamma_1 \overset{z_1}{\Rightarrow} \cdots$$

such that each $r_i \overset{z_i}{\Rightarrow} \gamma_{i+1}$ is provable according to the rules with respect to $\mathcal{M}$. We say a derivation *terminates* if it is finite and the last γ is terminal. A partial derivation is an initial segment of a complete derivation.

Example 12. Given the model below,



according to the SOS for $\langle a, s \rangle$ and the non-deterministic action in the model, the only two derivations of a at s are:

$$\langle a, s \rangle \overset{a}{\Rightarrow} s \text{ and } \langle a, s \rangle \overset{a}{\Rightarrow} w.$$

Let $\pi_1 = \texttt{while } p \texttt{ do } a$. An infinite derivation of π_1 at s is:

$$\langle \pi_1, s \rangle \overset{\epsilon}{\Rightarrow} \langle \texttt{if } p \texttt{ then } (a; \pi_1) \texttt{ else } \epsilon, s \rangle \overset{\epsilon}{\Rightarrow} \langle a; \pi_1, s \rangle \overset{a}{\Rightarrow} \langle \pi_1, s \rangle \overset{\epsilon}{\Rightarrow} \cdots$$

and a finite derivation of π_1 at s is:

$$\begin{aligned} \langle \pi_1, s \rangle \overset{\epsilon}{\Rightarrow} \quad & \langle \texttt{if } p \texttt{ then } (a; \pi_1) \texttt{ else } \epsilon, s \rangle \overset{\epsilon}{\Rightarrow} \langle a; \pi_1, s \rangle \overset{a}{\Rightarrow} \langle \pi_1, w \rangle \overset{\epsilon}{\Rightarrow} \\ & \langle \texttt{if } p \texttt{ then } (a; \pi_1) \texttt{ else } \epsilon, w \rangle \overset{\epsilon}{\Rightarrow} \langle \epsilon, w \rangle \overset{\epsilon}{\Rightarrow} w. \end{aligned}$$

Let $\pi = (\pi_1; b) = (\texttt{while } p \texttt{ do } a); b$. A finite derivation at s is:

$$\begin{aligned} \langle \pi, s \rangle \overset{\epsilon}{\Rightarrow} \quad & \langle (\texttt{if } p \texttt{ then } (a; \pi_1) \texttt{ else } \epsilon); b, s \rangle \overset{\epsilon}{\Rightarrow} \langle (a; \pi_1); b, s \rangle \overset{a}{\Rightarrow} \langle \pi_1; b, w \rangle \overset{\epsilon}{\Rightarrow} \\ & \langle (\texttt{if } p \texttt{ then } (a; \pi_1) \texttt{ else } \epsilon); b, w \rangle \overset{\epsilon}{\Rightarrow} \langle \epsilon; b, w \rangle \overset{\epsilon}{\Rightarrow} \langle b; w \rangle \overset{b}{\Rightarrow} t. \end{aligned}$$

To define the semantics of know-how, we need a generalized notion of *strong executability* proposed in [51]. Intuitively, a program is strongly executable, if it can always successfully terminate, thus it never gets stuck.

[7] Sometimes there are extra conditions specified besides the rules, as in the cases of IfthenP and IfthenN.

Definition 13 *(Strong executability).* A program π is strongly executable at s in $\mathcal{M}$ if all the derivations from $\langle \pi, s\rangle$ terminate. We say π is strongly executable on $[s]$, if it is strongly executable at each $s' \in [s]$.

In Example 12, the program `while` p `do` a is not strongly executable at s since it has an infinite derivation which does not terminate.

Note that the ϵ-transitions do not change the states. Therefore, intuitively, complete traces of a program at s can be recovered from all the derivations by ignoring π_i in γ_i and $\stackrel{\epsilon}{\Rightarrow}$ transitions. For example, from $\langle \pi, s\rangle \stackrel{\epsilon}{\Rightarrow} \langle \pi', s\rangle \stackrel{a}{\Rightarrow} t$ we can recover the trace *sat*. Formally, we have:

Definition 14 *(Trace).* Given a finite or an infinite derivation $\langle \pi_0, s_0\rangle \stackrel{z_0}{\Rightarrow} \langle \pi_1, s_1\rangle \stackrel{z_1}{\Rightarrow} \cdots$ of π at s, we can define its non-ϵ subsequence of labels as $z_{j_1} z_{j_2} \cdots$. Then the *complete* trace according to the derivation is defined as the state-label mixed sequence:

$$s_0 z_{j_1} s_{j_1+1} z_{j_2} s_{j_2+1} \cdots$$

Given π and s, the set of all complete traces of π at s are the set induced by the set of all derivations of π at s. A *partial trace* of π at s is an initial segment of a complete trace of π at s, which can also be induced from a partial derivation. We also call a partial trace which is not complete a *non-complete trace*.

By the definition above, it is obvious that if $s_0 a_1 s_1 \cdots$ is a trace of π on s then $s_{i+1} \in Q(a_{i+1})(s_i)$ for all $i \geq 0$. We use $Q(\pi)(s)$ to denote the set of states t such that t is the last state of a finite complete trace of π on s. If S is a set of states, we then use $Q(\pi)(S)$ to mean $\bigcup_{s\in S} Q(\pi)(s)$, in particular $Q(\pi)([s]) = \bigcup_{u\sim s} Q(\pi)(u)$.

In Example 12, a finite complete trace of `while` p `do` a at s is saw, and $sasasw$ is another one, while an infinite trace is $sasasasa\cdots$. A partial trace of $(\texttt{while}\ p\ \texttt{do}\ a); b$ is $sasaw$, which is an initial segment of a complete trace $sasawbt$.

Remark 15. Note that an *infinite* derivation may have a *finite* complete trace, e.g., any derivation for `while` $\top$ `do` ϵ in a model, as it does not change the current state. However, according to our definition, this program is not strongly executable since it has an infinite derivation: you cannot finish executing the plan although you never move to a different state.

Below is a useful observation that we can always change the one-step partial derivations by replacing one successor by another.

Proposition 16. *Let $\pi \in \texttt{Prg}$, $s \in \mathcal{M}$, $a \in \mathbf{A}$ be given, and let $t' \in Q(a)(s)$, we have that*

- *If $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} \langle \pi', t\rangle$ for some t then $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} \langle \pi', t'\rangle$;*
- *If $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} t$ for some t then $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} t'$.*

Proof. By induction on π, we show that if $t' \in Q(a)(s)$ and $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} \gamma$, then $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} \langle \pi', t'\rangle$ if $\gamma = \langle \pi', t\rangle$, and $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} t'$ if $\gamma = t$. Now suppose $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} \gamma$, according to the rules, π cannot be in the form of ϵ, $\texttt{if}\ \varphi\ \texttt{then}\ \pi_1\ \texttt{else}\ \pi_2$, or $\texttt{while}\ \varphi\ \texttt{do}\ \pi'$.

For the basic case of $\pi \in \mathbf{A}$, we then have that $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} \gamma$ is derived from the rule Atom. it follows that $\pi = a$ and $\gamma = t$. Since $t' \in Q(a)(s)$, by the rule Atom again, we have that $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} t'$.

For the inductive step, there are two cases:

- The derivation $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} \gamma$ is derived from the rule CompoL. It follows that $\pi = \pi_1; \pi_2$, $\gamma = \langle \pi'; \pi_2, t\rangle$ and $\langle \pi_1, s\rangle \stackrel{a}{\Rightarrow} \langle \pi', t\rangle$. By IH, we have that $\langle \pi_1, s\rangle \stackrel{a}{\Rightarrow} \langle \pi', t'\rangle$. Thus, by the rule CompoL again, we have that $\langle \pi_1; \pi_2, s\rangle \stackrel{a}{\Rightarrow} \langle \pi'; \pi_2, t'\rangle$.
- The derivation $\langle \pi, s\rangle \stackrel{a}{\Rightarrow} \gamma$ is derived from the rule CompoR. It follows that $\pi = \pi_1; \pi_2$, $\gamma = \langle \pi_2, t\rangle$ and $\langle \pi_1, s\rangle \stackrel{a}{\Rightarrow} t$. By IH, we have that $\langle \pi_1, s\rangle \stackrel{a}{\Rightarrow} t'$. Thus, by the rule CompoR again, we have that $\langle \pi_1; \pi_2, s\rangle \stackrel{a}{\Rightarrow} \langle \pi_2, t'\rangle$. □

Based on the above proposition, we show that we can do the same replacements to the traces.

Proposition 17. *Let $\pi \in \texttt{Prg}$ be strongly executable on $[s]$, and let $s_0 \cdots s_n a_{n+1} s_{n+1}$ be a partial trace of π on s, we then have that for each $t \in Q(a_{n+1})(s_n)$, the sequence $s_0 \cdots s_n a_{n+1} t$ is also a partial trace of π on s.*

Proof. Please note that all derivations of π on $s' \in [s]$ are finite, since π is strongly executable on all states in $[s]$. Since $s_0 \cdots s_n a_{n+1} s_{n+1}$ is a partial trace of π on s, there is a partial derivation $\gamma_0 \stackrel{z_1}{\Rightarrow} \gamma_1 \stackrel{z_2}{\Rightarrow} \cdots \gamma_k \stackrel{z_{k+1}}{\Rightarrow} \gamma_{k+1}$ such that $\gamma_k = \langle \pi', s_n\rangle$, $z_{k+1} = a_{n+1}$ and the state in γ_{k+1} is s_{n+1}. By Proposition 16, we have that $\gamma_k \stackrel{z_{k+1}}{\Rightarrow} \gamma_{k+1}[t/s_{n+1}]$. Thus, $\gamma_0 \stackrel{z_1}{\Rightarrow} \gamma_1 \stackrel{z_2}{\Rightarrow} \cdots \gamma_k \stackrel{z_{k+1}}{\Rightarrow} \gamma_{k+1}[t/s_{n+1}]$ is also a partial derivation of π on s. Thus, the sequence $s_0 \cdots s_n a_{n+1} t$ is also a partial trace of π on s. □

Now we are ready to present our uniform treatment of the logic of knowing how.

3. Epistemic language of know-how and its semantics

In this section, we first define ten types of plans using fragments of Prg. Then we give a generic form of the semantics for an epistemic language of know-how based on these plans. We show that at the level of models, these different plans induce different notions of know-how. Finally we prove that the know-how logic is also invariant under the bisimulation we introduced earlier.

3.1. Ten types of plans

Definition 18. Inspired by various earlier work on know-how or planning in the literature, we define the following subtypes of plans (For abbreviations, we call them x-plans where $x \in \mathtt{PLAN}=\{\mathtt{S},\ \mathtt{L},\ \mathtt{P},\ \mathtt{T},\ \mathtt{U},\ \mathtt{C},\ \mathtt{K},\ \mathtt{F},\ \mathtt{UK},\ \mathtt{CK}\}$):

- Simple plans [39], i.e., either the empty plan or atomic actions:

  S-plan $\quad \sigma ::= \epsilon \mid a$

- Linear plans [51], i.e. finite sequences of atomic actions:

  L-plan $\quad \sigma ::= \epsilon \mid a \mid \sigma;\sigma$

- Skippable linear plans [48], i.e. linear plans with executability tests to skip:

  P-plan $\quad \sigma ::= \epsilon \mid \mathtt{if}\ \langle a\rangle\top\ \mathtt{then}\ a\ \mathtt{else}\ \epsilon \mid \sigma;\sigma$

- Stoppable linear plans [28], i.e. linear plans with executability tests to stop:

  T-plan $\quad \sigma ::= \epsilon \mid \mathtt{if}\ \langle a\rangle\top\ \mathtt{then}\ a;\sigma\ \mathtt{else}\ \epsilon$

- Unbounded linear plans [27], i.e., linear plans with while-loops:

  U-plan $\quad \sigma ::= \epsilon \mid a \mid \sigma;\sigma \mid \mathtt{while}\ \varphi\ \mathtt{do}\ \sigma$

- Conditional plans, i.e., linear plans with conditionals:

  C-plan $\quad \sigma ::= \epsilon \mid a \mid \sigma;\sigma \mid \mathtt{if}\ \varphi\ \mathtt{then}\ \sigma\ \mathtt{else}\ \sigma$

- Knowledge-based plans [22], i.e. programs with epistemic conditions[8]:

  K-plan $\quad \sigma ::= \epsilon \mid a \mid \sigma;\sigma \mid \mathtt{if}\ \mathcal{K}\varphi\ \mathtt{then}\ \sigma\ \mathtt{else}\ \sigma \mid \mathtt{while}\ \mathcal{K}\varphi\ \mathtt{do}\ \sigma$

- Full plans include simply any Prg programs [26]:

  F-plan $\quad \sigma ::= \epsilon \mid a \mid \sigma;\sigma \mid \mathtt{if}\ \varphi\ \mathtt{then}\ \sigma\ \mathtt{else}\ \sigma \mid \mathtt{while}\ \varphi\ \mathtt{do}\ \sigma$

where φ is an $\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ formula. When we restrict the formula in unbounded linear plans and conditional plans to be in the form of $\mathcal{K}\varphi$ as in knowledge-based plans, we then have *knowledge-based unbounded plans* (UK-plans) and *knowledge-based conditional plans* (CK-plans).

Linear plans are finite sequences of actions used widely in planning. A simple plan is a special one-step linear plan, such as those used in [39], if restricted to the single-agent setting. Linear plans are special cases of unbounded plans [27] that can be executed indefinitely. As discussed in [28], sometimes we want to make sure we have reached the goal whenever we cannot go further, and this is captured by the stoppable plans [28]. A skippable plan can be viewed as another weaker linear plan for which you can skip the non-executable actions [48]. Conditional plans add the branching structure to plans, as discussed in contingent planning. In many cases, it is reasonable to assume that the plan is based on knowledge conditions in order to be truly executable by the agent as in UK and CK plans. On the other hand, a general condition φ can also be viewed as a binary sensing action, e.g., the executability test $\langle a\rangle\top$ in skippable or stoppable plans.

[8] Note that as in the standard S5 epistemic logic, in our setting $\neg\mathcal{K}\varphi$ is equivalent to $\mathcal{K}\neg\mathcal{K}\varphi$, thus the knowledge conditions also essentially include the negations of $\mathcal{K}\varphi$ formulas.

3.2. The language and semantics of know-how

Based on these types of plans, we are ready to define the corresponding notions of know-how. Throughout the paper, we will be using the following language proposed in [16] to provide axiomatizations of logics of know-how:

Definition 19 *(ELKh Language).* Let $\mathbf{P}$ be a countable set of variables, the language $\mathcal{L}^{\mathsf{ELKh}}$ of epistemic logic with the know-how operator (ELKh) is defined as follows:

$$\varphi ::= \top \mid p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \mathcal{K}\varphi \mid \mathcal{K}h\varphi$$

where $p \in \mathbf{P}$.

Note that in this language we *do not* have any action modality $[a]$ of EAL, which is only used in the semantics for specifying the plans. See Section 6.1 for the discussions about the languages of EAL, ELKh, and Prg.

This simple language can express very interesting propositions about highly non-trivial planning such as: *knowing how to know whether* φ $(\mathcal{K}h(\mathcal{K}\varphi \vee \mathcal{K}\neg\varphi))$; *knowing how to achieve* φ *but then will not be able to turn it off* $(\mathcal{K}h\varphi \wedge \neg\mathcal{K}h(\varphi \wedge \mathcal{K}h\neg\varphi))$. Moreover, if we generalize the language to a multi-agent setting by attaching agent names to the modalities then we can express propositions such as *agent i knows how to make sure not only* φ *but also that agent j does not know how to achieve* $\neg\varphi$ $(\mathcal{K}h_i(\varphi \wedge \neg\mathcal{K}h_j\neg\varphi))$. See Section 6.3 for the discussions on the multi-agent generalization, and see Section 6.6 for using model checking to do planning in our framework.

The semantics of $\mathcal{L}^{\mathsf{ELKh}}$ is also based on the epistemic transition systems introduced in Definition 4. Intuitively, the agent knows how to achieve φ iff there *exists* a plan such that the agent *knows* that it is executable and it can guarantee φ in the end.

Definition 20 *(Semantics).* The truth conditions for basic propositions, Boolean connectives and the epistemic operator $\mathcal{K}$ are as before in the cases of EAL. We give a generic form of ten different semantics $\models_x$ for the know-how operator $\mathcal{K}h$ where x is one of PLAN=$\{$S, L, P, T, U, C, K, F, UK, CK$\}$.

$\mathcal{M}, s \models_x \mathcal{K}h\varphi$	$\iff$	there is an x-plan σ such that for all $s' \in [s]$ 1. σ is strongly executable on s'; 2. $\mathcal{M}, t \models_x \varphi$ for each $t \in Q(\sigma)(s')$.

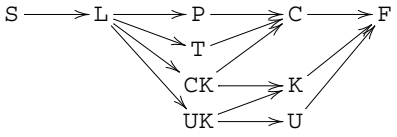
Clearly, the above semantics is equivalent to the following form which is easier to use in the technical discussions in the rest of the paper.

$\mathcal{M}, s \models_x \mathcal{K}h\varphi$	$\iff$	there is an x-plan σ such that: 1. σ is strongly executable on $[s]$; 2. $\mathcal{M}, t \models_x \varphi$ for each $t \in Q(\sigma)([s])$.

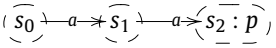
We can compare the variants of the $\mathcal{K}h$ semantics by defining a natural ordering.

Definition 21 *(Expressivity ordering for $\mathcal{K}h$).* For any $x, y \in$ PLAN, we write $x \to y$ if for all the pointed models $(\mathcal{M}, s)$, for any propositional letter p, $\mathcal{M}, s \models_x \mathcal{K}hp$ implies $\mathcal{M}, s \models_y \mathcal{K}hp$.

It is clear that the relation $\to$ is a partial order. By definitions of the plans and the truth conditions of the semantics for $\mathcal{K}h$, we have the following picture of the $\to$ relation:



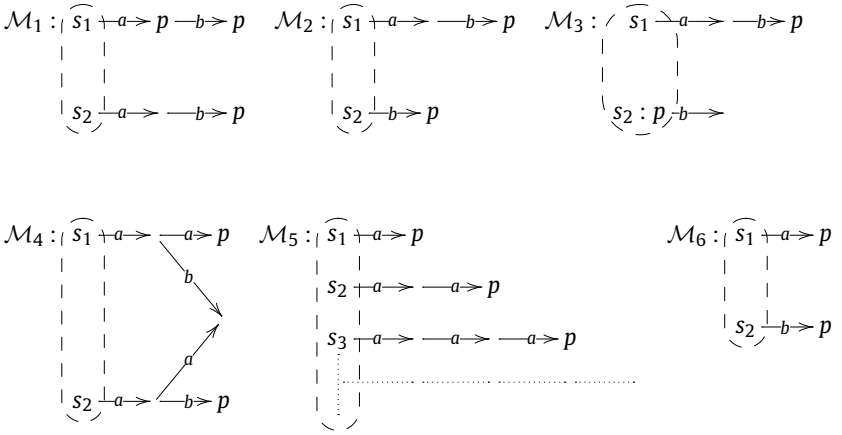
Note that the above relations do not hold if we generalize the condition on $\mathcal{K}hp$ to $\mathcal{K}h\varphi$ in Definition 21, since φ may contain negative formulas such as $\neg\mathcal{K}hp$. For example, $\mathcal{M}_0, s_0 \models_{\mathrm{S}} \mathcal{K}h\neg\mathcal{K}hp$ (by the ϵ plan) but $\mathcal{M}_0, s_0 \not\models_{\mathrm{F}} \mathcal{K}h\neg\mathcal{K}hp$ where $\mathcal{M}_0$ is depicted below:



Moreover, we can show the ordering above is mostly *strict*.

Proposition 22. *Except for* $\mathtt{U} \to \mathtt{F}$ *and* $\mathtt{UK} \to \mathtt{K}$*, the arrows in the above graph are strict in the sense that if* $x \to y$ *then it is* not *the case that* $y \to x$*.*

Proof. To differentiate these $\vDash_x$, consider the following models $\mathcal{M}_{1\text{-}6}$ (the epistemic relations are omitted for the non-initial states):



- It is not hard to check that $\mathcal{M}_1, s_1 \nvDash_{\mathtt{S}} \mathcal{K}hp$, but $\mathcal{M}_1, s_1 \vDash_{\mathtt{L}} \mathcal{K}hp$. Therefore $\mathtt{L} \not\to \mathtt{S}$.
- Due to the skippable linear plan $\mathtt{if}\ \langle a\rangle\top\ \mathtt{then}\ a\ \mathtt{else}\ \epsilon; b$, we then have that $\mathcal{M}_2, s_1 \vDash_{\mathtt{P}} \mathcal{K}hp$. Since $\mathcal{M}_2, s_1 \nvDash_{\mathtt{L}} \mathcal{K}hp$, we then have that $\mathtt{P} \not\to \mathtt{L}$.
- Due to the stoppable linear plan $\mathtt{if}\ \langle a\rangle\top\ \mathtt{then}\ (a; \mathtt{if}\ \langle b\rangle\top\ \mathtt{then}\ b\ \mathtt{else}\ \epsilon)\ \mathtt{else}\ \epsilon$, we have that $\mathcal{M}_3, s_1 \vDash_{\mathtt{T}} \mathcal{K}hp$. Since $\mathcal{M}_3, s_1 \nvDash_{\mathtt{L}} \mathcal{K}hp$, this means that $\mathtt{T} \not\to \mathtt{L}$.
- Due to the conditional knowledge-based plan $a; \mathtt{if}\ \mathcal{K}[a]p\ \mathtt{then}\ a\ \mathtt{else}\ b$, we have that $\mathcal{M}_4, s_1 \vDash_{\mathtt{CK}} \mathcal{K}hp$. Since $\mathcal{M}_4, s_1 \nvDash_{\mathtt{L}} \mathcal{K}hp$, this means that $\mathtt{CK} \not\to \mathtt{L}$.
- Due to the unbounded knowledge-based plan $\mathtt{while}\ \mathcal{K}\neg\mathcal{K}p\ \mathtt{do}\ a$, $\mathcal{M}_5, s_1 \vDash_{\mathtt{UK}} \mathcal{K}hp$. Since $\mathcal{M}_5, s_1 \nvDash_{\mathtt{L}} \mathcal{K}hp$, this means that $\mathtt{UK} \not\to \mathtt{L}$.
- It is not hard to check that $\mathcal{M}_2, s_1 \nvDash_x \mathcal{K}hp$ for all $x \in \{\mathtt{T}, \mathtt{CK}, \mathtt{UK}\}$,[9] and that $\mathcal{M}_3, s_1 \nvDash_x \mathcal{K}hp$ for all $x \in \{\mathtt{P}, \mathtt{CK}, \mathtt{UK}\}$, and that $\mathcal{M}_4, s_1 \nvDash_x \mathcal{K}hp$ for all $x \in \{\mathtt{P}, \mathtt{T}\}$, and that $\mathcal{M}_5, s_1 \nvDash_x \mathcal{K}hp$ for all $x \in \{\mathtt{P}, \mathtt{T}, \mathtt{CK}\}$. This means that each pair in $\{\mathtt{P}, \mathtt{T}, \mathtt{CK}, \mathtt{UK}\}$ except $(\mathtt{CK}, \mathtt{UK})$ is incomparable. This also implies that $\mathtt{C} \not\to \mathtt{x}$ for all $x \in \{\mathtt{P}, \mathtt{T}, \mathtt{CK}\}$. Since $\mathcal{M}_5, s_1 \vDash_{\mathtt{UK}} \mathcal{K}hp$ but $\mathcal{M}_5, s_1 \nvDash_{\mathtt{CK}} \mathcal{K}hp$, this implies that $\mathtt{K} \not\to \mathtt{CK}$.
- Due to the unbounded linear plan $\mathtt{while}\ \langle a\rangle\top\ \mathtt{do}\ a; \mathtt{while}\ \langle b\rangle\top\ \mathtt{do}\ b$, we have that $\mathcal{M}_6, s_1 \vDash_{\mathtt{U}} \mathcal{K}hp$. Since $\mathcal{M}_6, s_1 \nvDash_{\mathtt{UK}}$, this means that $\mathtt{U} \not\to \mathtt{UK}$.
- Due to the plan $\mathtt{if}\ \langle a\rangle\top\ \mathtt{then}\ a\ \mathtt{else}\ b$, we have that $\mathcal{M}_6, s_1 \vDash_{\mathtt{C}} \mathcal{K}hp$. Due to $\mathcal{M}_5, s_1 \vDash_{\mathtt{UK}} \mathcal{K}hp$, we have that $\mathcal{M}_5, s_1 \vDash_{\mathtt{K}} \mathcal{K}hp$. Since $\mathcal{M}_5, s_1 \nvDash_{\mathtt{C}} \mathcal{K}hp$ and $\mathcal{M}_6, s_1 \nvDash_{\mathtt{K}} \mathcal{K}hp$, this implies that $\mathtt{C}$ and $\mathtt{K}$ are incomparable. Thus $\mathtt{F} \not\to \mathtt{C}$ and $\mathtt{F} \not\to \mathtt{K}$. □

We conjecture that $\mathtt{U} \to \mathtt{F}$ and $\mathtt{UK} \to \mathtt{K}$ are also strict but leave it for a future occasion.

Based on the above observations, we know that these variants of the semantics for $\mathcal{K}h$ are indeed quite different. Now a natural question to ask is what are the know-how logics with respect to each of them?

Surprisingly, the answer is that they all share the *same* logic in terms of the valid formulas! Before proving it in the next section, we define the following two simple properties shared by all the above-mentioned semantics of $\mathcal{K}h$, by which we will prove a general completeness result.

Proposition 23. *For each* $x \in \mathtt{PLAN}$*, the semantics* $\vDash_x$ *satisfies the following two* $\mathcal{K}h$*-properties for any pointed model* $\mathcal{M}, s$*:*

1. *If* $\mathcal{M}, s \vDash_x \mathcal{K}h\varphi$*, then there is a program* $\pi \in \mathtt{Prg}$ *such that* π *is strongly executable on* $[s]$*, and* $\mathcal{M}, t \vDash_x \varphi$ *for each* $t \in Q(\pi)([s])$*;*
2. *If there is a simple plan* σ *such that* σ *is strongly executable on* $[s]$*, and* $\mathcal{M}, t \vDash_x \varphi$ *for each* $t \in Q(\sigma)([s])$*, then* $\mathcal{M}, s \vDash_x \mathcal{K}h\varphi$*.*

Proof. Condition 1 trivially holds since every $\vDash_x$ semantics is based on a sublanguage of the full plans $(\mathtt{Prg})$.

[9] For the semantics based on knowledge-based plans, it is impossible to do different actions on the states that are in the same epistemic equivalence class.

For condition 2, we only need to consider two kinds of semantics that do not include the simple plans (a) *prima facie*, i.e., $\models_{\mathtt{T}}$ and $\models_{\mathtt{P}}$. However, by the generic semantics of $\mathcal{K}h$, if a simple plan a can achieve φ then both the skippable plan $\mathtt{if}\ \langle a\rangle\top\ \mathtt{then}\ a\ \mathtt{else}\ \epsilon$ and the stoppable plan $\mathtt{if}\ \langle a\rangle\top\ \mathtt{then}\ a;\epsilon\ \mathtt{else}\ \epsilon$ will do so. □

The above two properties will play a crucial role in the following completeness proof.[10]

3.3. ELKh is invariant under bisimulation

Recall the notion of bisimulation for EAL that we introduced in Definition 6. To conclude this section, we prove that ELKh is also invariant under bisimulation with respect to all the $\models_x$. Clearly the crucial step is to show $\mathcal{K}h\varphi$ is invariant under bisimulation. For this, we first need to show the strong executability is preserved under bisimulation. We need some technical preparations below.

Proposition 24. *Given a model $\mathcal{M}$ and a program π, if $\langle\pi, s\rangle \overset{z}{\Rightarrow} \gamma$ and $\mathcal{M}, s \leftrightarrows \mathcal{M}, s'$, then there is some state t' such that $\langle\pi, s'\rangle \overset{z}{\Rightarrow} \gamma[t'/t]$ and $\mathcal{M}, t \leftrightarrows \mathcal{M}, t'$, where t is the state in γ.*

Proof. We prove it by induction on π.

- $\pi := \epsilon$. We have that $\langle\pi, s\rangle \overset{z}{\Rightarrow} \gamma$ is derived by the rule Skip. It follows that $z = \epsilon$ and $\gamma = t = s$. By Skip, we have that $\langle\pi, s'\rangle \overset{\epsilon}{\Rightarrow} s'$. It is obvious that $\mathcal{M}, s \leftrightarrows \mathcal{M}, s'$.
- $\pi := a$ for some $a \in \mathbf{A}$. We then have that $\langle\pi, s\rangle \overset{z}{\Rightarrow} \gamma$ is derived by the rule Atom. It follows that $z = a$, $\gamma = t$, and $t \in Q(a)(s)$. Since $\mathcal{M}, s \leftrightarrows \mathcal{M}, s'$, from this follows that there exists some state t' such that $t' \in Q(a)(s')$ and $\mathcal{M}, t \leftrightarrows \mathcal{M}, t'$. Since $t' \in Q(a)(s')$, by the rule Atom, we have that $\langle a, s'\rangle \overset{a}{\Rightarrow} t'$.
- $\pi := \pi_1; \pi_2$. There are two cases: either $\langle\pi, s\rangle \overset{z}{\Rightarrow} \gamma$ is derived by the rule CompoL, or by the rule CompoR.
  If it is derived by CompoL, we then have that $\gamma = \langle\pi_1'; \pi_2, t\rangle$ and $\langle\pi_1, s\rangle \overset{z}{\Rightarrow} \langle\pi_1', t\rangle$. By IH, we then have that there exists some state t' such that $\langle\pi_1, s'\rangle \overset{z}{\Rightarrow} \langle\pi_1', t'\rangle$ and $\mathcal{M}, t \leftrightarrows \mathcal{M}, t'$. Because of $\langle\pi_1, s'\rangle \overset{z}{\Rightarrow} \langle\pi_1', t'\rangle$, by CompoL, we have that $\langle\pi_1; \pi_2, s'\rangle \overset{z}{\Rightarrow} \langle\pi_1'; \pi_2, t'\rangle$.
  If it is derived by CompoR, we then have that $\gamma = \langle\pi_2, t\rangle$ and $\langle\pi_1, s\rangle \overset{z}{\Rightarrow} t$. By IH, we then have that there exists some state t' such that $\langle\pi_1, s'\rangle \overset{z}{\Rightarrow} t'$ and $\mathcal{M}, t \leftrightarrows \mathcal{M}, t'$. Because of $\langle\pi_1, s'\rangle \overset{z}{\Rightarrow} t'$, by CompoR, we have that $\langle\pi_1; \pi_2, s'\rangle \overset{z}{\Rightarrow} \langle\pi_2, t'\rangle$.
- $\pi := \mathtt{if}\ \varphi\ \mathtt{then}\ \pi_1\ \mathtt{else}\ \pi_2$ where $\varphi \in \mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$. There are two cases: either $\langle\pi, s\rangle \overset{z}{\Rightarrow} \gamma$ is derived by the rule IfthenP, or by the rule IfthenN.
  If it is derived by IfthenP, we then have that $\gamma = \langle\pi_1, s\rangle$ and $\mathcal{M}, s \models \varphi$. Since $\mathcal{M}, s \leftrightarrows \mathcal{M}, s'$, by Proposition 7, we have that $\mathcal{M}, s' \models \varphi$. By IfthenP, we then have that $\langle\pi, s'\rangle \overset{z}{\Rightarrow} \langle\pi_1, s'\rangle$.
  If it is derived by IfthenN, we then have that $\gamma = \langle\pi_2, s\rangle$ and $\mathcal{M}, s \not\models \varphi$. Since $\mathcal{M}, s \leftrightarrows \mathcal{M}, s'$, by Proposition 7, we have that $\mathcal{M}, s' \not\models \varphi$. By IfthenN, we then have that $\langle\pi, s'\rangle \overset{z}{\Rightarrow} \langle\pi_2, s'\rangle$.
- $\pi := \mathtt{while}\ \varphi\ \mathtt{do}\ \pi'$ where $\varphi \in \mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$. We then have that $\langle\pi, s\rangle \overset{z}{\Rightarrow} \gamma$ is derived by While. It follows that $z = \epsilon$ and $\gamma = \langle\mathtt{if}\ \varphi\ \mathtt{then}\ (\pi'; \mathtt{while}\ \varphi\ \mathtt{do}\ \pi')\ \mathtt{else}\ \epsilon, s\rangle$. By While, we have that $\langle\pi, s'\rangle \overset{\epsilon}{\Rightarrow} \langle\mathtt{if}\ \varphi\ \mathtt{then}\ (\pi'; \mathtt{while}\ \varphi\ \mathtt{do}\ \pi')\ \mathtt{else}\ \epsilon, s'\rangle$. □

Proposition 25. *Let $\mathcal{M}, s_0 \leftrightarrows \mathcal{M}, t_0$. If $\gamma_0 \overset{z_1}{\Rightarrow} \cdots \gamma_n$ is a derivation of π on s_0 and the state in γ_i is s_i for all $0 \le i \le n$, then there are states $t_0, \cdots, t_n$ such that $\gamma_0[t_0/s_0] \overset{z_1}{\Rightarrow} \cdots \gamma_n[t_n/s_n]$ is a derivation of π on t_0 and $\mathcal{M}, s_i \leftrightarrows \mathcal{M}, t_i$ for all $0 \le i \le n$.*

Proof. We prove it by induction on n. For the case of $n = 0$, let γ_0 be $\langle\pi, s_0\rangle$. This means that $\langle\pi, s_0\rangle$ gets stuck. If $\langle\pi, t_0\rangle$ does not get stuck, then there exist z and γ' such that $\langle\pi, t_0\rangle \overset{z}{\Rightarrow} \gamma'$. Since $\mathcal{M}, s_0 \leftrightarrows \mathcal{M}, t_0$, by Proposition 24, we have that there exists some state s' such that $\langle\pi, s_0\rangle \overset{z}{\Rightarrow} \gamma'[s'/t']$ where t' is the state in γ'. Contradiction! Therefore, $\langle\pi, t_0\rangle$ also gets stuck. Thus, $\langle\pi, t_0\rangle$ is a derivation of π on t_0.

Assuming the induction hypothesis (IH) for n, for the case of $n+1$, we have that $\gamma_0 \overset{z_1}{\Rightarrow} \cdots \gamma_{n+1}$ is a derivation of π on s_0. Please note that the states in γ_i is s_i for all $0 \le i \le n+1$. We then have that $\gamma_1 \overset{z_1}{\Rightarrow} \cdots \gamma_{n+1}$ is a derivation on s_1. Let γ_0 be $\langle\pi, s_0\rangle$. Since $\langle\pi, s_0\rangle \overset{z_1}{\Rightarrow} \gamma_1$ and $\mathcal{M}, s_0 \leftrightarrows \mathcal{M}, t_0$, by Proposition 24, there exists some state t_1 such that $\langle\pi, t_0\rangle \overset{z_1}{\Rightarrow} \gamma_1[t_1/s_1]$ and $\mathcal{M}, s_1 \leftrightarrows \mathcal{M}, t_1$. Thus, by IH, we have that there are $t_1 \cdots t_{n+1}$ such that $\gamma_1[t_1/s_1] \overset{z_2}{\Rightarrow} \cdots \gamma_{n+1}[t_{n+1}/s_{n+1}]$ is a derivation

[10] In fact, we can define a much larger class of alternative semantics satisfying these two properties, and show they also induce the same logic of know-how.

on t_1 and $\mathcal{M}, s_i \leftrightarrows \mathcal{M}, t_i$ for all $1 \leq i \leq n+1$. Since $\langle \pi, t_0 \rangle \overset{z_1}{\Rightarrow} \gamma_1[t_1/s_1]$, we have that $\gamma_0[t_0/s_0] \overset{z_1}{\Rightarrow} \cdots \gamma_{n+1}[t_{n+1}/s_{n+1}]$ is a derivation of π on t_0. □

Proposition 26. *Given a program $\pi \in \mathtt{Prg}$, if $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [t]$, we then have that π is strongly executable on $[s]$ iff it is strongly executable on $[t]$.*

Proof. Assume towards contradiction that π is strongly executable on $[s]$ but not on $[t]$. It follows that there is some $t' \in [t]$ such that π is not strongly executable on t'. We then have that there is a derivation of π on t' which does not terminate. Since $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [t]$, we have that $\mathcal{M}, s' \leftrightarrows \mathcal{M}, t'$ for some $s' \in [s]$.

If the derivation of π on t' which does not terminate gets stuck, then the length of the derivation is finite. Since $\mathcal{M}, s' \leftrightarrows \mathcal{M}, t'$, by Proposition 25, we have that there is a derivation of π on s' which also gets stuck. Contradiction! Therefore, there are no derivation of π on t' which gets stuck.

If the derivation of π on t' which does not terminate is infinite, let it be $\langle \pi, t' \rangle \overset{z_1}{\Rightarrow} \gamma_1 \cdots$, and let the state in γ_i is t_i for all $i \geq 1$. Let the set X be $\{\tau_i := \langle \pi, s' \rangle \overset{z_1}{\Rightarrow} \gamma_1[s_1/t_1] \cdots \gamma_i[s_i/t_i] \mid i \in \mathbb{N}$, and $\mathcal{M}, s_j \leftrightarrows \mathcal{M}, t_j$ for all $0 \leq j \leq i\}$. (please note that τ_0 is $\langle \pi, s' \rangle$, so $X \neq \emptyset$.) We define a binary relation R on X such that $(\tau, \tau') \in R$ iff τ is a proper initial segment of τ'. By Proposition 24, we know that R is an entire relation. Thus, by the Axiom of Dependence Choice, we have that there is an infinite derivation of π on s'. This is contradictory to the fact that π is strongly executable on s'. Thus, there is no derivation of π on t' which is infinite.

Thus, we have shown that all derivations of π on t' terminate. This is contradictory to our assumption that π is not strongly executable on t'. Therefore, π is strongly executable on $[t]$ if π is strongly executable on $[s]$. Similarly, we can show that π is strongly executable on $[s]$ if π is strongly executable on $[t]$. □

Now we show that executing a strongly executable plan on bisimilar models will result in bisimilar states.

Proposition 27. *Let π be strongly executable on $[s]$. If $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [s']$, we then have that for each $t \in Q(\pi)([s])$, there is $t' \in Q(\pi)([s'])$ such that $\mathcal{M}, t \leftrightarrows \mathcal{M}, t'$.*

Proof. Since π is strongly executable on all states in $[s]$, every derivation of π on each state in $[s]$ is finite. Let $t \in Q(\pi)(s_0)$ for some $s_0 \in [s]$. It follows that there is a finite derivation $\gamma_0 \overset{z_1}{\Rightarrow} \cdots \gamma_n$ of π on s_0 such that t is the state in γ_n. Since $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [s']$ and $s_0 \in [s]$, $\mathcal{M}, s_0 \leftrightarrows \mathcal{M}, t_0$ for some $t_0 \in [s']$. Since $\gamma_0 \overset{z_1}{\Rightarrow} \cdots \gamma_n$ is a derivation of π on s_0, by Proposition 25, we have that there is some t' such that $t' \in Q(\pi)(s')$ and $\mathcal{M}, t \leftrightarrows \mathcal{M}, t'$. □

Finally, we can show that $\mathcal{L}^{\mathsf{ELKh}}$ is invariant under bisimulation. This result will help us in the discussions in Section 5.

Theorem 28. *For each $x \in \mathtt{PLAN}$, if $\mathcal{M}, s \leftrightarrows \mathcal{M}, s'$, then $\mathcal{M}, s \vDash_x \varphi$ iff $\mathcal{M}, s' \vDash_x \varphi$ for all formula φ in $\mathcal{L}^{ELKh}$.*

Proof. By induction on φ, we will show that $\mathcal{M}, s \vDash_x \varphi$ iff $\mathcal{M}, s' \vDash_x \varphi$ for all formula φ in $\mathcal{L}^{\mathsf{ELKh}}$. Next, we will only focus on the case of $\mathcal{K}h\varphi$; the other cases are straightforward.

If $\mathcal{M}, s \vDash_x \mathcal{K}h\varphi$, then there is an x-plan θ such that θ is strongly executable on $[s]$ and that $\mathcal{M}, t \vDash_x \varphi$ for each $t \in Q(\theta)([s])$. It follows by $\mathcal{M}, s \leftrightarrows \mathcal{M}, s'$ that $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [s']$. By Proposition 26, we then have that θ is strongly executable on $[s']$. For each $t' \in Q(\theta)([s'])$, it follows by Proposition 27 that there is some $t \in Q(\theta)([s])$ such that $\mathcal{M}, t \leftrightarrows \mathcal{M}, t'$. Since $\mathcal{M}, t \vDash_x \varphi$ for each $t \in Q(\theta)([s])$, we then by IH have that $\mathcal{M}, t' \vDash_x \varphi$ for each $t' \in Q(\theta)([s'])$. It follows that $\mathcal{M}, s' \vDash_x \mathcal{K}h\varphi$. Thus, we have shown that if $\mathcal{M}, s \vDash_x \mathcal{K}h\varphi$ then $\mathcal{M}, s' \vDash_x \mathcal{K}h\varphi$. Similarly, it can be shown that if $\mathcal{M}, s' \vDash_x \mathcal{K}h\varphi$ then $\mathcal{M}, s \vDash_x \mathcal{K}h\varphi$. □

4. A unified axiomatization

In this section, we show that all the 10 semantics induce exactly the same know-how logic over arbitrary epistemic transition systems.

4.1. The proof system $\mathbb{SLKH}$

Given the semantics, first note that some normal modal logic axioms are *not* valid.

Proposition 29. *The following are not valid with respect to any $\vDash_x$:*

- $\mathcal{K}hp \wedge \mathcal{K}hq \rightarrow \mathcal{K}h(p \wedge q)$
- $\mathcal{K}h(p \rightarrow q) \wedge \mathcal{K}hp \rightarrow \mathcal{K}hq$

For the invalidity of the first formula: $\mathcal{K}hp$ and $\mathcal{K}hq$ can be witnessed by two different plans without a unified plan to make sure $(p \wedge q)$. For the invalidity of the K "axiom" for $\mathcal{K}h$, notice that $\mathcal{K}h\neg p$ implies $\mathcal{K}h(p \to q)$. However $\mathcal{K}h\neg p$ and $\mathcal{K}hp$ together in this case does not say anything about the plan for q.

Therefore the know-how logic is non-normal, as well-known in the literature. We propose the following proof system $\mathbb{SLKH}$:

Axioms

TAUT	all axioms of propositional logic
DISTK	$\mathcal{K}p \wedge \mathcal{K}(p \to q) \to \mathcal{K}q$
T	$\mathcal{K}p \to p$
4	$\mathcal{K}p \to \mathcal{K}\mathcal{K}p$
5	$\neg\mathcal{K}p \to \mathcal{K}\neg\mathcal{K}p$
AxKtoKh	$\mathcal{K}p \to \mathcal{K}hp$
AxKhtoKKh	$\mathcal{K}hp \to \mathcal{K}\mathcal{K}hp$
AxKhbot	$\neg\mathcal{K}h\bot$

Rules

$$\text{MP}\ \frac{\varphi, \varphi \to \psi}{\psi} \qquad \text{NECK}\ \frac{\varphi}{\mathcal{K}\varphi}$$

$$\text{MONOKh}\ \frac{\varphi \to \psi}{\mathcal{K}h\varphi \to \mathcal{K}h\psi} \qquad \text{SUB}\ \frac{\varphi}{\varphi[\psi/p]}$$

Theorem 30. *The proof system $\mathbb{SLKH}$ is sound with respect to $\vDash_x$ for all $x \in$ PLAN.*

Proof. The validity of axioms T, 4, 5 is due to the standard semantics for $\mathcal{K}$. For the validity of other axioms, we just need to use the two properties in Proposition 23 shared by all the $\vDash_x$ semantics.

The axiom AxKtoKh says if p is known then you know how to achieve p. It is validity is guaranteed by Kh-property (2) where we essentially allow the empty plan. The axiom AxKhbot says we cannot guarantee contradiction, which indirectly requires that the plan should terminate at someplace. Finally, the axiom AxKhtoKKh is the positive introspection axiom for $\mathcal{K}h$ whose validity is due to condition (1). □

Note that the negative introspection for know-how is also derivable.

Proposition 31. $\vdash \neg\mathcal{K}hp \to \mathcal{K}\neg\mathcal{K}hp$.

Proof.

(1)	$\neg\mathcal{K}\mathcal{K}hp \to \neg\mathcal{K}hp$	AxKhtoKKh
(2)	$\mathcal{K}\neg\mathcal{K}\mathcal{K}hp \to \mathcal{K}\neg\mathcal{K}hp$	(1), NECK
(3)	$\neg\mathcal{K}\mathcal{K}hp \to \mathcal{K}\neg\mathcal{K}\mathcal{K}hp$	5
(4)	$\neg\mathcal{K}\mathcal{K}hp \to \mathcal{K}\neg\mathcal{K}hp$	(2), (3), MP
(5)	$\neg\mathcal{K}hp \to \neg\mathcal{K}\mathcal{K}hp$	T
(6)	$\neg\mathcal{K}hp \to \mathcal{K}\neg\mathcal{K}hp$	(4), (5), MP □

Remark 32. Compared with the know-how logic in [16], two axioms are missing: the composition axiom $\mathcal{K}h\mathcal{K}h\varphi \to \mathcal{K}h\varphi$, and the axiom $\mathcal{K}h\varphi \to \mathcal{K}h\mathcal{K}\varphi$. The composition axiom will come back in the next section when we restrict ourselves to finite models (see discussions there for its invalidity over arbitrary models). The other axiom depends on some basic assumption about the agent,[11] which requires some extra property on the models (see Section 6.4 for further discussions).

4.2. Completeness

Next, we will show that the proof system $\mathbb{SLKH}$ is complete with respect to knowing-how logics that satisfies the two properties in Proposition 23. Firstly, we introduce some notations. Given a set of formulas Δ, let $\Delta|_{\mathcal{K}} = \{\mathcal{K}\varphi \mid \mathcal{K}\varphi \in \Delta\}$, $\Delta|_{\neg\mathcal{K}} = \{\neg\mathcal{K}\varphi \mid \neg\mathcal{K}\varphi \in \Delta\}$, $\Delta|_{\mathcal{K}h} = \{\mathcal{K}h\varphi \mid \mathcal{K}h\varphi \in \Delta\}$, and $\Delta|_{\neg\mathcal{K}h} = \{\neg\mathcal{K}h\varphi \mid \neg\mathcal{K}h\varphi \in \Delta\}$.

Definition 33 *(Canonical model).* The canonical model $\mathcal{M}^c = \langle W, \mathbf{A}^c, \sim, \{Q(a) \mid a \in \mathbf{A}^c\}, V\rangle$ is defined as:

- $W = \{(\Delta, i) \mid \Delta \subseteq \mathcal{L}^{\mathsf{ELKh}}$ is a maximal consistent set in $\mathbb{SLKH}$, and $i \in \mathbb{N}\}$;

[11] For example, I know how to get drunk, but I may not know how to know that I am drunk, assuming that one cannot realize he or she is drunk when really drunk.

- $\mathbf{A}^c = \{(\varphi, i) \mid \varphi \in \mathcal{L}^{\mathsf{ELKh}}, i \in \mathbb{N}\}$;
- $(\Delta, i) \sim (\Gamma, j) \iff i = j$ and $\Delta|_{\mathcal{K}} = \Gamma|_{\mathcal{K}}$;
- for each $(\varphi, i) \in \mathbf{A}^c$, $Q(\varphi, i) = \{((\Delta, j), (\Gamma, k) \mid \mathcal{K}h\varphi \in \Delta$, and if $i = j$ then $\varphi \in \Gamma\}$
- for each $p \in \mathbf{P}$, $p \in V(\Delta, i) \iff p \in \Delta$.

Please note that, by Definition 33, $Q(\varphi, i)(\Delta, j) = W$ if $\mathcal{K}h\varphi \in \Delta$ and $i \neq j$. For a state $(\Delta, i) \in W$, we also write it as Δ^i, and similarly, we write $(\varphi, i) \in \mathbf{A}^c$ as φ^i. For example, let Δ_0 and Δ_1 be two maximal consistent sets of $\mathbb{SLKH}$, and let i be a natural number. We will use Δ_0^i and Δ_1^i to denote (Δ_0, i) and (Δ_1, i) respectively.

The existence lemma for $\mathcal{K}$ is routine.

Proposition 34. *Let Δ be a maximal consistent set. If $\neg\mathcal{K}\varphi \in \Delta$ then there exists a maximal consistent set Γ such that $\Delta|_{\mathcal{K}} = \Gamma|_{\mathcal{K}}$ and $\neg\varphi \in \Gamma$.*

Proof. Let $D = \Delta|_{\mathcal{K}} \cup \Delta|_{\neg\mathcal{K}} \cup \{\neg\varphi\}$. Firstly, we show that D is consistent. If D is not consistent, then there are $\mathcal{K}\psi_1 \cdots \mathcal{K}\psi_n \in \Delta|_{\mathcal{K}}$, and $\neg\mathcal{K}\chi_1 \cdots \neg\mathcal{K}\chi_m \in \Delta|_{\neg\mathcal{K}}$ such that $\vdash \mathcal{K}\psi_1 \wedge \cdots \wedge \mathcal{K}\psi_n \wedge \neg\mathcal{K}\chi_1 \wedge \cdots \wedge \neg\mathcal{K}\chi_m \to \varphi$. By the Rule NECK and axiom DISTK, we have that $\vdash \mathcal{K}\mathcal{K}\psi_1 \wedge \cdots \wedge \mathcal{K}\mathcal{K}\psi_n \wedge \mathcal{K}\neg\mathcal{K}\chi_1 \wedge \cdots \wedge \mathcal{K}\neg\mathcal{K}\chi_m) \to \mathcal{K}\varphi$. Since $\mathcal{K}\psi_1 \cdots \mathcal{K}\psi_n, \neg\mathcal{K}\chi_1 \cdots \neg\mathcal{K}\chi_m \in \Delta$, by the Axioms T and 4 and that Δ is maximal consistent, we have that $\mathcal{K}\mathcal{K}\psi_1 \cdots \mathcal{K}\mathcal{K}\psi_n, \mathcal{K}\neg\mathcal{K}\chi_1 \cdots \mathcal{K}\neg\mathcal{K}\chi_m \in \Delta$. It follows that $\mathcal{K}\varphi \in \Delta$. Contradiction! Thus, we have that D is consistent. By Lindenbaum's lemma, we then have that there is a maximal consistent set Γ such that $D \subseteq \Gamma$. It follows that $\Delta|_{\mathcal{K}} \subseteq \Gamma|_{\mathcal{K}}$ and $\neg\varphi \in \Gamma$. We then only need to show that $\Gamma|_{\mathcal{K}} \subseteq \Delta|_{\mathcal{K}}$. Assume that there is $\mathcal{K}\psi \in \Gamma$ but $\mathcal{K}\psi \notin \Delta$. Since Δ is maximal consistent, $\neg\mathcal{K}\psi \in \Delta$. Since $\Delta|_{\neg\mathcal{K}} \subseteq D \subseteq \Gamma$, we then have that $\neg\mathcal{K}\psi \in \Gamma$. This is contradictory to $\mathcal{K}\psi \in \Gamma$ and that Γ is consistent. Therefore, we have that $\Gamma|_{\mathcal{K}} \subseteq \Delta|_{\mathcal{K}}$. □

To show the truth lemma for $\mathcal{K}h$, we need some preparation.

Proposition 35. *Given $\Delta^i \in W$, if $\varphi \in \Gamma$ for each $\Gamma^i \in [\Delta^i]$, then $\mathcal{K}h\varphi \in \Delta$.*

Proof. Firstly, we will show that $\mathcal{K}\varphi \in \Delta$. If $\mathcal{K}\varphi \notin \Delta$, then $\neg\mathcal{K}\varphi \in \Delta$. By Proposition 34, we then have that there exists a maximal consistent set Γ such that $\neg\varphi \in \Gamma$ and $\Gamma|_{\mathcal{K}} = \Delta|_{\mathcal{K}}$. By the Definition 33, we then have that $\Gamma^i \sim \Delta^i$. Contradiction! Thus, we have that $\mathcal{K}\varphi \in \Delta$. By the Axiom AxKtoKh, we then have that $\mathcal{K}h\varphi \in \Delta$. □

Proposition 36. *Let $\psi^i \in \mathbf{A}^c$ be executable on $\Delta^i \in W$. If $\varphi \in \Gamma$ for each $\Gamma^j \in Q(\psi^i)(\Delta^i)$, then we have that $\mathcal{K}h\varphi \in \Delta$.*

Proof. Since ψ^i is executable on Δ^i, there is some $\Theta^k \in W$ such that $(\Delta^i, \Theta^k) \in Q(\psi^i)$. By Definition 33, we then have that $\mathcal{K}h\psi \in \Delta$. Next, we will show that $\{\psi, \neg\varphi\}$ is not consistent. If it is, there exists a maximal consistent set Φ such that $\{\psi, \neg\varphi\} \subseteq \Phi$. We then have that, for each $k \in \mathbb{N}$, $\Phi^k \in Q(\psi^i)(\Delta^i)$. Since $\neg\varphi \in \Phi$, this is contradictory to that $\varphi \in \Gamma$ for each $\Gamma^j \in Q(\psi^i)(\Delta^i)$. Thus, we have that $\{\psi, \neg\varphi\}$ is not consistent. It follows that $\vdash \psi \to \varphi$. By the Rule MONOKh, we have that $\vdash \mathcal{K}h\psi \to \mathcal{K}h\varphi$. Since $\mathcal{K}h\psi \in \Delta$, $\mathcal{K}h\varphi \in \Delta$. □

Proposition 37. *Let π be strongly executable on Δ^i. The sequence $\Delta_0^{i_0} \cdots \Delta_j^{i_j} a_{i_{j+1}} \Delta_{j+1}^{i_{j+1}}$, where $Q(a_{i_{j+1}})(\Delta_j^{i_j}) = W$, is a partial trace of π on Δ^i such that $\Delta^i = \Delta_0^{i_0}$, and $\varphi \in \Gamma$ for each $\Gamma^j \in Q(\pi)(\Delta^i)$. Then we have that $\vdash \varphi \leftrightarrow \top$.*

Proof. Since π is strongly executable on Δ^i, all derivations of π on Δ^i are finite. Moreover, we know that $\vdash \varphi \leftrightarrow \top$ iff φ is in each maximal consistent set. Suppose towards contradiction that there is some maximal consistent set Γ_0 such that $\varphi \notin \Gamma_0$, i.e. $\neg\varphi \in \Gamma_0$, we will show that it will lead to an infinite derivation of π on Δ^i. The core idea is simple: we can step-by-step extend the derivation behind the partial trace in the statement of the proposition, and make sure that at any step, the extension not be a complete derivation by using Γ_0 which includes $\neg\varphi$, based on the fact that all the complete derivations will end up with some φ state.

Let $X = \{(\tau, a) \mid \tau := \gamma_0 \overset{z_0}{\Rightarrow} \cdots \gamma_n$ is a partial derivation of π on Δ^i, there exists some γ such that $\gamma_n \overset{a}{\Rightarrow} \gamma$, and $Q(a)(s_{\gamma_n}) = W$, where s_{γ_n} is the state in $\gamma_n\}$. Since $\Delta_0^{i_0} \cdots \Delta_j^{i_j} a_{i_{j+1}} \Delta_{j+1}^{i_{j+1}}$ is a partial trace of π on Δ^i and $Q(a_{i_{j+1}})(\Delta_j^{i_j}) = W$, we have that $X \neq \emptyset$. Let R be a binary relation on X, which is defined as $(\tau, a)R(\tau', a')$ iff $\tau' := \tau \overset{a}{\Rightarrow} \gamma_0 \overset{\epsilon}{\Rightarrow} \gamma_1 \overset{\epsilon}{\Rightarrow} \cdots \overset{\epsilon}{\Rightarrow} \gamma_k$ for some $\gamma_0 \cdots \gamma_k$ where $k \in \mathbb{N}$. If R is an entire binary relation, i.e., every element in X has an R successor, then because of $X \neq \emptyset$, it follows by the Axiom of Dependent Choice that there is an infinite derivation of π on Δ^i.

Please note that since π is a string with finite length, there are only finitely many primitive actions occurring in π. Let them be $\varphi_1^{m_1}, \cdots, \varphi_n^{m_n} \in \mathbf{A}^c$, and let l_0 be a natural number not in $\{m_1, \cdots, m_n\}$.

Next, we will show that R is an entire binary relation on X. Given $(\tau, a) \in X$ where $\tau := \gamma_0 \cdots \gamma_n$, we have that τ is a partial derivation of π on Δ^i, $Q(a)(s_{\gamma_n}) = W$, and $\gamma_n \overset{a}{\Rightarrow} \gamma$ for some γ. Thus, by Proposition 16, we have that $\gamma_n \overset{a}{\Rightarrow} \gamma[\Gamma_0^l / s_\gamma]$ for each $l \in \mathbb{N}$. Therefore, for each $l \in \mathbb{N}$, $\tau \overset{a}{\Rightarrow} \gamma[\Gamma_0^l / s_\gamma]$ is a partial derivation of π on Δ^i. Since $\varphi \notin \Gamma_0$ and $\varphi \in \Gamma$ for each

$\Gamma^j \in Q(\pi)(\Delta^i)$, these follow that, for each $l \in \mathbb{N}$, $\tau \overset{a}{\Rightarrow} \gamma[\Gamma_0^l/s_\gamma]$ is not a complete derivation of π on Δ^i. In other words, the partial derivation $\tau \overset{a}{\Rightarrow} \gamma[\Gamma_0^l/s_\gamma]$ needs to be extended. It might be extended by $\overset{\epsilon}{\Rightarrow}$, like $\tau \overset{a}{\Rightarrow} \gamma[\Gamma_0^l/s_\gamma] \overset{\epsilon}{\Rightarrow} \gamma'$. However, since $\overset{\epsilon}{\Rightarrow}$ cannot change the states, the state in γ' is still Γ_0^l. For the same reason, the partial derivation $\tau \overset{a}{\Rightarrow} \gamma[\Gamma_0^l/s_\gamma] \overset{\epsilon}{\Rightarrow} \gamma'$ needs to be extended continually. Furthermore, it cannot be extended by $\overset{\epsilon}{\Rightarrow}$ infinitely many times, since π is strongly executable on Δ^i. Therefore, after it is extended by $\overset{\epsilon}{\Rightarrow}$ with some $k \in \mathbb{N}$ times for $\tau \overset{a}{\Rightarrow} \gamma[\Gamma_0^{l_0}/s_\gamma]$, $\tau' := \tau \overset{a}{\Rightarrow} \gamma[\Gamma_0^{l_0}/s_\gamma] \overset{\epsilon}{\Rightarrow} \gamma_1' \overset{\epsilon}{\Rightarrow} \cdots \gamma_k'$ will be extended by $\overset{z}{\Rightarrow}$ where $x := \varphi_j^{m_j}$ is some action in $\{\varphi_1^{m_1}, \cdots, \varphi_n^{m_n}\}$. Please note that $s_{\gamma_k'} = \Gamma_0^{l_0}$. By the definition of the canonical model $\mathcal{M}^c$, we have that $Q(\varphi_j^{m_j})(\Gamma_0^{l_0}) = W$. Therefore, we have that $(\tau', \varphi_j^{m_j}) \in X$. Since $(\tau, a)R(\tau', \varphi_j^{m_j})$, R is an entire binary relation on X.

In sum, we have shown that R is an entire binary relation on X. Therefore, by the Axiom of Dependent Choice, there is an infinite derivation of π on Δ^i because of $X \neq \emptyset$. This is contradictory to the fact that π is strongly executable on Δ^i. Please note that if π is strongly executable on Δ^i then all derivations of π on Δ^i are finite. Therefore, we have that φ is in every maximal consistent set, i.e. $\vdash \varphi \leftrightarrow \top$. □

Proposition 38. *Let π be strongly executable on Δ^i. The sequence $\Delta_0^{i_0} a_1 \Delta_1^{i_1} a_2 \cdots \Delta_n^{i_n}$ where $n \geq 1$ is a trace of π on Δ^i, and $\varphi \in \Gamma$ for each $\Gamma^m \in Q(\pi)(\Delta^i)$. Then we have that $\mathcal{K}h\varphi \in \Delta$.*

Proof. Let a_1 be some $\psi^j \in \mathbf{A}^c$. If $j \neq i$, it follows by the definition of the canonical model $\mathcal{M}^c$ that $Q(\Delta^i)(\psi^j) = W$. By Proposition 37, we have that $\vdash \varphi \leftrightarrow \top$. By the rule `MONOKh` we have $\vdash \mathcal{K}h\varphi \leftrightarrow \mathcal{K}h\top$. By Rule `NECK` and Axiom `AxKtoKh`, we have that $\vdash \mathcal{K}h\top$. Thus, we have $\vdash \mathcal{K}h\varphi$, and then $\mathcal{K}h\varphi \in \Delta$.

Next, we will show that if $j = i$ we then also have $\mathcal{K}h\varphi \in \Delta$. Firstly by Proposition 17, for each $\Gamma^k \in Q(a_1)(\Delta^i)$, $\Delta^i a_1 \Gamma^k$ is a partial trace of π on Δ^i. If $j = i$, there are two cases: either all partial traces $\Delta^i a_1 \Gamma^k$ where $\Gamma^k \in Q(a_1)(\Delta^i)$ are complete traces of π on Δ^i, or not.

- If all traces $\Delta^i a_1 \Gamma^k$ where $\Gamma^k \in Q(a_1)(\Delta^i)$ are complete traces of π on Δ^i, then $Q(a_1)(\Delta^i) \subseteq Q(\pi)(\Delta^i)$. Since $\varphi \in \Gamma$ for each $\Gamma^m \in Q(\pi)(\Delta^i)$, $\varphi \in \Gamma$ for each $\Gamma^k \in Q(a_1)(\Delta^i)$. By Proposition 36, we have that $\mathcal{K}h\varphi \in \Delta$.
- If there is some $\Gamma^k \in Q(a_1)(\Delta^i)$ such that $\Delta^i a_1 \Gamma^k$ is a non-complete trace of π on Δ^i, then the set $D = \{(\Gamma^k, \chi^l) \mid \Gamma^k \in Q(a_1)(\Delta^i), \chi^l \in \mathbf{A}^c$, and there exists $\Theta^m \in W$ such that $\Delta^i a_1 \Gamma^k \chi^l \Theta^m$ is a possible trace of π on $\Delta^i\}$ is not empty. Moreover, let $ran(D) = \{\chi^l \mid (\Gamma^k, \chi^l) \in D$ for some $\Gamma^k\}$. We then have that each $\chi^l \in ran(D)$ must occur in π. Since π is a finite string, we then have that $ran(D)$ is a finite set.
  If there is some $(\Gamma^k, \chi^l) \in D$ such that $k \neq l$, by the definition of $\mathcal{M}^c$, we have that $Q(\chi^l)(\Gamma^k) = W$. By Proposition 37, we then have that $\vdash \varphi \leftrightarrow \top$, and then it is easy to show that $\mathcal{K}h\varphi \in \Delta$. If, for each $(\Gamma^k, \chi^l) \in D$, we have that $k = l$. Let $m \in \mathbb{N}$ be a natural number that is bigger than all the superscript in $ran(D)$. Note that this is possible since $ran(D)$ is finite. By the definition of the set D. We then have that, for each $(\Gamma^k, \chi^l) \in D$, $\Delta^i a_1 \Gamma^m$ is a complete trace of π on Δ^i, i.e. $\Gamma^m \in Q(\pi)(\Delta^i)$. Therefore, we have that $\varphi \in \Gamma$. In other words, we have shown that, for each $\Gamma^k \in Q(\Delta^i)(a_1)$, we also have that $\varphi \in \Gamma$, even though the trace $\Delta^i a_1 \Gamma^k$ is a non-complete trace of π on Δ^i. Thus, we have that $\varphi \in \Gamma$ for each $\Gamma^k \in Q(\Delta^i)(a_1)$. Therefore, by Proposition 36, we have that $\mathcal{K}h\varphi \in \Delta$. □

Now we are ready to prove the truth lemma.

Lemma 39. *For each $\mathcal{L}^{ELKh}$ formula φ, $\mathcal{M}^c, \Delta^i \vDash_x \varphi$ iff $\varphi \in \Delta$ for any $x \in PLAN$.*

Proof. We prove it by induction on φ using the two properties of $\vDash_x$ as in Proposition 23. The atomic and Boolean cases are straightforward. The case of $\mathcal{K}\varphi$ can be proved by Proposition 34. Next, we only focus on the case of $\mathcal{K}h\varphi$.

Right to Left: If $\mathcal{K}h\varphi \in \Delta$, firstly, we show that φ is consistent. Supposing it is not, we then have $\vdash \varphi \to \bot$. By Rule `MONOKh`, it follows that $\vdash \mathcal{K}h\varphi \to \mathcal{K}h\bot$. Now by $\mathcal{K}h\varphi \in \Delta$ and Axiom `AxKhbot`, Δ is not consistent, which contradicts to the assumption that Δ is a consistent set. Thus, we show that φ is consistent. By Lindenbaum's lemma, we have that there is some maximal consistent set Γ such that $\varphi \in \Gamma$.

Next, we will show that the action $\varphi^i \in \mathbf{A}^c$ is simply a good plan to witness the truth of $\mathcal{K}h\varphi$. Since $\mathcal{K}h\varphi \in \Delta$, by the Axioms `AxKhtoKKh` and `T`, we have that $\mathcal{K}h\varphi \in \Delta_0$ for each $\Delta_0^i \in [\Delta^i]$. By Definition 33, we then have that $\Delta_0^i \xrightarrow{\varphi^i} \Gamma^0$ for each $\Delta_0^i \in [\Delta^i]$. Thus a is strongly executable on $[\Delta^i]$. What is more, by Definition 33, we also have that $\varphi \in \Gamma_0$ for each $\Gamma_0^j \in Q(a)([\Delta^i])$. By IH, it follows that $\mathcal{M}^c, \Gamma_0^j \vDash_x \varphi$ for each $\Gamma_0^j \in Q(a)([\Delta^i])$. Therefore, by the second condition of Proposition 23, we have that $\mathcal{M}^c, \Delta^i \vDash_x \mathcal{K}h\varphi$.

Left to Right: If $\mathcal{M}^c, \Delta^i \vDash_x \mathcal{K}h\varphi$, by the first property of Proposition 23, we then have that there is some `Prg` π such that π is strongly executable on $[\Delta^i]$ and that $\mathcal{M}^c, \Gamma^j \vDash_x \varphi$ for all $\Gamma^j \in Q(\pi)([\Delta^i])$. By IH, we have that $\varphi \in \Gamma$ for all $\Gamma^j \in Q(\pi)([\Delta^i])$.

Please note that, for each $\Delta_0^i \in [\Delta^i]$, the set of traces of π on Δ_0^i cannot be empty, since π is strongly executable on $[\Delta^i]$. Therefore, there are two cases: (1.) there is some $\Delta_0^i \in [\Delta^i]$ such that the length of whose trace is no less than 1; (2.)

for each $\Delta_0^i \in [\Delta^i]$, all traces of π on Δ_0^i do not change the state, i.e. $Q(\pi)([\Delta^i]) = [\Delta^i]$. For (1.), by Proposition 38, we have that $\mathcal{K}h\varphi \in \Delta$. For (2.), since $\varphi \in \Gamma$ for all $\Gamma^j \in Q(\pi)([\Delta^i])$, by Proposition 34 we have that $\mathcal{K}\varphi \in \Delta$. By the Axiom AxKtoKh, we have $\mathcal{K}h\varphi \in \Delta$. □

Based on the truth lemma, the completeness is immediate.

Theorem 40 *(Completeness). The proof system $\mathbb{SLKH}$ is strongly complete with respect to $\models_x$ for all $x \in$ PLAN.*

Therefore, by Proposition 23, we have

Corollary 41. *The proof system $\mathbb{SLKH}$ is sound and strongly complete with respect to $\models_x$ for all $x \in$ PLAN. Thus for all $\varphi \in \mathcal{L}^{ELKh}$, $\models_x \varphi \iff \models_y \varphi$ for all $x \in$ PLAN.*

4.3. Decidability

In this section, we show that our know-how logic is decidable by proving some version of the finite model property.

Let Φ be a subformula-closed set of formulas. Below we define the closure of Φ, and use it to build a filtration model with respect to Φ.

Definition 42. The closure $cl(\Phi)$ is $\Phi \cup \{\mathcal{K}\varphi, \neg\mathcal{K}\varphi \mid \varphi \in \Phi\}$.

Definition 43. Let $\mathcal{M} = \langle W, \mathbf{A}, \sim, \{Q(a) \mid a \in \mathbf{A}\}, V\rangle$ be a model, and let Φ be a subformula closed set of formulas. Given $x \in$ PLAN, let $\leftrightsquigarrow$ be the relation on the states of $\mathcal{M}$ defined by:

$$s \leftrightsquigarrow t \iff \text{for all } \varphi \in cl(\Phi) : (\mathcal{M}, s \models_x \varphi \text{ iff } \mathcal{M}, t \models_x \varphi).$$

Note that $\leftrightsquigarrow$ is an equivalence relation. We denote the equivalence class of a state s of $\mathcal{M}$ with respect to $\leftrightsquigarrow$ by $|s|$. The model $\mathcal{M}^f = \langle W^f, \mathbf{A}^\Phi, \sim, \{Q(a) \mid a \in \mathbf{A}^\Phi\}, V^f\rangle$ is defined as

- $W^f = \{|s| \mid s \in W\}$;
- $\mathbf{A}^\Phi = \{\varphi \mid \mathcal{K}h\varphi \in \Phi\}$;
- $|s| \sim |t| \iff$ for all $\mathcal{K}\varphi \in cl(\Phi) : \mathcal{M}, s \models_x \mathcal{K}\varphi$ iff $\mathcal{M}, t \models_x \mathcal{K}\varphi$;
- for all $\varphi \in \mathbf{A}^\Phi$, $(|s|, |t|) \in Q(\varphi)$ iff $\mathcal{M}, s \models_x \mathcal{K}h\varphi$ and $\mathcal{M}, t \models_x \varphi$;
- $p \in V^f(|s|)$ iff $p \in V(s)$ for all $p \in \Phi$.

By the definition above, it is obvious that $|s| \sim |t|$ if $s \sim t$.

Now we can show the filtration will preserve the truth value with respect to $\models_S$ semantics.

Lemma 44. *For all $\varphi \in \Phi$, we have that $\mathcal{M}, s \models_x \varphi \iff \mathcal{M}^f, |s| \models_S \varphi$ for all $x \in$ PLAN.*

Proof. We prove it by induction on φ. Next, we will only focus on the cases of $\mathcal{K}\varphi$ and $\mathcal{K}h\varphi$ since the other cases are straightforward. Note that we will use the shared form of the semantics $\models_x$ in the following proof.

- $\mathcal{K}\varphi$. **Left to Right**: Assume that $\mathcal{M}, s \models_x \mathcal{K}\varphi$ but $\mathcal{M}^f, |s| \not\models_S \mathcal{K}\varphi$. Since $\mathcal{M}^f, |s| \not\models_S \mathcal{K}\varphi$, we have that $\mathcal{M}^f, |t| \not\models_S \varphi$ for some $|t|$ such that $|s| \sim |t|$. By IH, we then have that $\mathcal{M}, t \not\models_x \varphi$. We then have that $\mathcal{M}, t \not\models_x \mathcal{K}\varphi$. Since $|s| \sim |t|$, by Definition 43, we have that $\mathcal{M}, s \not\models_x \mathcal{K}\varphi$. Contradiction! Thus, if $\mathcal{M}, s \models_x \mathcal{K}\varphi$ then $\mathcal{M}^f, |s| \models_S \mathcal{K}\varphi$.
  Right to Left: Assume that $\mathcal{M}^f, |s| \models_S \mathcal{K}\varphi$ but $\mathcal{M}, s \not\models_x \mathcal{K}\varphi$. Since $\mathcal{M}, s \not\models_x \mathcal{K}\varphi$, we have that $\mathcal{M}, t \not\models_x \varphi$ for some t such that $s \sim t$. By IH, we have $\mathcal{M}^f, |t| \not\models_S \varphi$. Since $s \sim t$, we then have that $|s| \sim |t|$. Thus, we have that $\mathcal{M}^f, |s| \not\models_S \mathcal{K}\varphi$. Contradiction! Thus, if $\mathcal{M}^f, |s| \models_S \mathcal{K}\varphi$ then $\mathcal{M}, s \models_x \mathcal{K}\varphi$.
- $\mathcal{K}h\varphi$. **Left to Right**: Assume that $\mathcal{M}, s \models_x \mathcal{K}h\varphi$. We will show that $\mathcal{M}^f, |s| \models_S \mathcal{K}h\varphi$. Let $a = \varphi$. Please note that $a \in \mathbf{A}^\Phi$. Moreover, we have that $\mathcal{K}\mathcal{K}h\varphi \in cl(\Phi)$, since $\mathcal{K}h\varphi \in \Phi$. Because of $\mathcal{M}, s \models_x \mathcal{K}h\varphi$, by the generic form of the semantics $\models_x$, we have that $\mathcal{M}, s \models_x \mathcal{K}\mathcal{K}h\varphi$. For each $|u|$ such that $|s| \sim |u|$, by Definition 43, we have that $\mathcal{M}, u \models_x \mathcal{K}\mathcal{K}h\varphi$. Therefore, $\mathcal{M}, u \models_x \mathcal{K}h\varphi$. Moreover, by the strong executability requirement in the semantics, there is some t such that $\mathcal{M}, t \models_x \varphi$, since $\mathcal{M}, s \models_x \mathcal{K}h\varphi$. Thus, we have that $(|u|, |t|) \in Q(a)$ for each $|u| \in \big[|s|\big]$ by the definition of Q. From this follows that a is executable on each $|u| \in \big[|s|\big]$. For each $|v| \in Q(a)(\big[|s|\big])$, by Definition 43, we have that $\mathcal{M}, v \models_x \varphi$. By IH, we then have that $\mathcal{M}^f, |v| \models_S \varphi$ for each $|v| \in Q(a)(\big[|s|\big])$. Therefore, we have that $\mathcal{M}^f, |s| \models_S \mathcal{K}h\varphi$.
  Right to Left: Assume that $\mathcal{M}^f, |s| \models_S \mathcal{K}h\varphi$. Next, we will show that $\mathcal{M}, s \models_x \mathcal{K}h\varphi$. Since $\mathcal{M}^f, |s| \models_S \mathcal{K}h\varphi$, there is a simple plan $z \in \mathbf{A}^\Phi \cup \{\epsilon\}$ such that z is strongly executable on $\big[|s|\big]$ and $\mathcal{M}^f, |t| \models_S \varphi$ for each $|t| \in Q(z)(\big[|s|\big])$. By IH, we have that $\mathcal{M}, t \models_x \varphi$ for each $|t| \in Q(z)(\big[|s|\big])$. Please note that $s \sim u$ implies $|s| \sim |u|$.

If $z = \epsilon$, we have that $|u| \in Q(z)(|s|)$ for each $u \in [s]$. Therefore, $\mathcal{M}, u \vDash_x \varphi$ for each $u \in [s]$. Thus, we have that $\mathcal{M}, s \vDash_x \mathcal{K}h\varphi$.
If $z \in \mathbf{A}^\Phi$, let z be some $\psi \in \mathbf{A}^\Phi$. Since z is strongly executable on $\big[|s|\big]$, from this follows that z is executable on $|s|$. By Definition 43, it follows that $\mathcal{M}, s \vDash_x \mathcal{K}h\psi$. Firstly, we show that $\psi \to \varphi$ holds on all states of $\mathcal{M}$. If it does not, there is some state w of $\mathcal{M}$ such that $\mathcal{M}, w \vDash_x \psi$ and $\mathcal{M}, w \nvDash_x \varphi$. By IH, we have that $\mathcal{M}^f, |w| \nvDash_{\mathbb{S}} \varphi$. Since $\mathcal{M}, w \vDash_x \psi$, by Definition 43, we have that $|s| \xrightarrow{\psi} |w|$. Since $\mathcal{M}^f, |w| \nvDash_{\mathbb{S}} \varphi$, this is contradictory to that $\mathcal{M}^f, |t| \vDash_{\mathbb{S}} \varphi$ for each $|t| \in Q(z)(\big[|s|\big])$. Therefore, we have that $\psi \to \varphi$ holds on all states of $\mathcal{M}$. Since $\mathcal{M}, s \vDash_x \mathcal{K}h\psi$, there exists an x-plan θ such that θ is strongly executable on $[s]$ and $\mathcal{M}, v \vDash_x \psi$ for each $v \in Q(\theta)(s)$. Since $\psi \to \varphi$ holds on all states of $\mathcal{M}$, we have that $\mathcal{M}, v \vDash_x \varphi$ for each $v \in Q(\theta)(s)$. Therefore, we have that $\mathcal{M}, s \vDash_x \mathcal{K}h\varphi$. □

The above lemma also gives us a version of small model property: if φ has a model with respect to $\vDash_x$ then it has a small finite filtration model with respect to $\vDash_{\mathbb{S}}$ (by setting Φ as the set of all subformulas of φ). Then it is immediate to have the decidability of the logic.

Theorem 45 *(Decidability). The logic $\mathbb{SLKH}$ is decidable.*

Remark 46. Note that what we showed in Lemma 44 is *not* the usual finite model property: note that the finite model of φ is with respect to the specific semantics $\vDash_{\mathbb{S}}$. It is not the case that for each $x \in \mathtt{PLAN}$ if $\varphi \in \mathcal{L}^{\mathsf{ELKh}}$ is $\vDash_x$-satisfiable then it is $\vDash_x$-satisfiable in a finite model. As we will show in the next section, with respect to $\vDash_{\mathtt{CK}}$, some $\mathcal{L}^{\mathsf{ELKh}}$-formula is valid only over finite models but not in general.

5. Conditional know-how over finite models

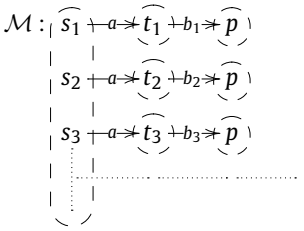
In [16], the axiomatization of the logic of strategically know-how includes the following crucial axiom saying that knowing how to know how implies knowing how:

$$\mathtt{AxKhKh}: \quad \mathcal{K}h\mathcal{K}h\psi \to \mathcal{K}h\psi$$

This axiom essentially captures the *compositionality* of the strategy-like plans in [16]. It says that if there is a plan σ to make sure that you have further plans to achieve ψ eventually, then you have a unified plan to achieve ψ. Note that after executing σ you may end up with various equivalence classes with respect to the epistemic relation, though on each of them you know how to achieve φ. Therefore, the idea of the unified plan is simple: first, do σ and then continue with the right plan depending on the epistemic state that you end up with. In [16], we can always have a plan to achieve this since the plans there are simply functions from epistemic equivalence classes to actions. However, in the current setting, plans are no longer arbitrary functions anymore. They are finite syntactic objects in $\mathtt{Prg}$. Whether a unified conditional plan is possible depends on the expressiveness of the plan specification language and the number of branching conditions we need to handle.

Intuitively, making the axiom $\mathtt{AxKhKh}$ valid requires the plan to be at least including the conditional plans. For example, it is easy to see that for linear plans, the axiom $\mathtt{AxKhKh}$ is not valid. However, the axiom $\mathtt{AxKhKh}$ is not valid even when we consider the full plan with the full expressive power in the conditions.

Example 47 *($\mathtt{AxKhKh}$ is not valid with respect to $\vDash_{\mathtt{F}}$).* Consider the model below: initially you do not know where you are but doing a can eliminate the uncertainty. However, depending on the resulting state t_k you need to do b_k to achieve p.



It is clear that $\mathcal{M}, s_1 \vDash_{\mathtt{F}} \mathcal{K}h\mathcal{K}hp$, since a is a good plan to make sure $\mathcal{K}hp$. However, to define a unified plan we need to do different things on infinitely many different states but the specification language cannot express infinitely many different conditions. Therefore $\mathcal{M}, s_1 \nvDash_{\mathtt{F}} \mathcal{K}hp$.

The above example also suggests a mismatch between the specification language of the plans and the information in the model: we cannot really express infinitely branching plans in the plan specification language. Now, what about restoring the balance between what can be specified and what is there in the model? This motivates us to consider the know-how logic over *finite models*.

Another motivation for considering finite models is to stay closer to potential AI applications, where practically the relevant models are finite. To be realistic, we will also only consider *knowledge-based plans* in this section such that an agent can execute it based on its own knowledge, although all the technical results can carry over to full plans without the constraint on the epistemic conditions.

Before our discussion on the axiomatization of know-how logic over finite models, we can simplify the setting a little bit further.

Recall that a knowledge-based plan (K-plan) is a full plan with epistemic conditions and its while-loop-free part is the conditional knowledge-based plans (CK-plans):

$$\theta ::= \epsilon \mid a \mid \theta; \theta \mid \texttt{if } \mathcal{K}\varphi \texttt{ then } \theta \texttt{ else } \theta$$

In the following, we show that we can just consider CK plans instead of the full knowledge-based plans since the while loops can be eliminated given a concrete finite model (cf. also [22]).

Proposition 48. *Given a finite pointed model* $(\mathcal{M}, s)$, $\mathcal{M}, s \vDash_{\mathrm{K}} \varphi \iff \mathcal{M}, s \vDash_{\mathrm{CK}} \varphi$.

Proof. (Sketch) A simple inductive proof will do the job. The only non-trivial inductive case is to show that: $\mathcal{M}, s \vDash_{\mathrm{K}} \mathcal{K}h\varphi \Longrightarrow \mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h\varphi$. Suppose $\mathcal{M}, s \vDash_{\mathrm{K}} \varphi$, then by the definition of $\vDash_{\mathrm{K}}$, there is a knowledge-based plan σ in Prg such that for all $s' \in [s]$

1 σ is strongly executable on s';
2 $\mathcal{M}, t \vDash_{\mathrm{K}} \varphi$ for each $t \in Q(\sigma)(s')$.

Based on the induction hypothesis, we just need to find a conditional plan σ' such that $Q(\sigma)(s) = Q(\sigma')(s)$ and σ' is strongly executable on all $s' \in [s]$. Namely, we need a while-loop free plan which functions just like σ.

The idea to find the while-free σ' is simple: in a finite model, if $\texttt{while } \varphi \texttt{ do } \pi$ always terminates (due to strong executability), then it must terminate at the same states within n repetitions of the conditional program $(\texttt{if } \varphi \texttt{ then } \pi \texttt{ else } \epsilon)$, where n is the size of the model. To see it, note that if a derivation from $\langle \texttt{while } \varphi \texttt{ do } \pi, s\rangle$ goes back to itself then it will never terminate. Thus it can only pass $\langle \texttt{while } \varphi \texttt{ do } \pi, t\rangle$ for a new state t each time when it repeats the loop according to the SOS rules. However, there are only finitely many states. Therefore, it is not hard to show that we can safely replace each $\langle \texttt{while } \varphi \texttt{ do } \pi\rangle$ recursively by $(\texttt{if } \varphi \texttt{ then } \pi \texttt{ else } \epsilon)^n$ to eliminate all the while-loops in the program. □

Based on the above proposition, $\vDash_{\mathrm{K}}$ and $\vDash_{\mathrm{CK}}$ are equivalent over finite models. In the following, we will axiomatize the know-how logic with respect to $\vDash_{\mathrm{CK}}$.

5.1. Extending $\mathbb{SLKH}$ with a composition axiom

We define a proof system $\mathbb{SLKHC}$ extending the system $\mathbb{SLKH}$ with the following axiom of composition.

$$\texttt{AxKhKhG}: \quad \mathcal{K}h(\varphi \vee \mathcal{K}h\psi) \to \mathcal{K}h(\varphi \vee \psi)$$

Note that the AxKhKh can be viewed as a special case of AxKhKhG when φ is $\bot$. However, we do need the generality of AxKhKhG, but leave the proof of the incompleteness of AxKhKh on top of $\mathbb{SLKH}$ to a future occasion.

As in [16], although the composition axiom looks intuitive, the proof for its validity is highly non-trivial. The rough idea is the following. To show that the composition axiom is valid, we need to show that if $\mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h(\varphi \vee \mathcal{K}h\psi)$ then $\mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h(\varphi \vee \psi)$ for each pointed model $(\mathcal{M}, s)$. By $\mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h(\varphi \vee \mathcal{K}h\psi)$, we know that there is a conditional knowledge-based plan σ such that $\mathcal{M}, t \vDash_{\mathrm{CK}} \varphi \vee \mathcal{K}h\psi$ for each $t \in Q(\sigma)([s])$. Let D be the set $\{t \in Q(\sigma)([s]) \mid \mathcal{M}, t \nvDash_{\mathrm{CK}} \varphi\}$. We then have that $\mathcal{M}, t \vDash_{\mathrm{CK}} \mathcal{K}h\psi$ for each $t \in D$. Therefore, for each $t \in D$, there is a conditional knowledge-based plan σ_t that witnesses $\mathcal{M}, t \vDash_{\mathrm{CK}} \mathcal{K}h\psi$. To show $\mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h(\varphi \vee \psi)$, we need to combine σ and all σ_t where $t \in D$. We need to unite all σ_t where $t \in D$ as one non-deterministic plan like "if φ_{t_1} then σ_{t_1} else (if φ_{t_2} then σ_{t_2} else ...)". This is possible since D is finite. Moreover, we need a formula φ_t to characterize t for each $t \in D$. On the other hand, if there is no formula to distinguish t and t', we need to show that σ_t is also a witness plan for $\mathcal{M}, t' \vDash_{\mathrm{CK}} \mathcal{K}h\psi$.

To show the validity of AxKhKhG, we first need some technical preparations.

Observation Note that without the derivation rule While, all the provable $\langle \pi, s\rangle \overset{z}{\Rightarrow} \gamma$ will only reduce the complexity of π or eliminate it completely, from left to right. With this observation, we can show that the derivation of a sequential composition of plans is essentially the sequential composition of the derivations for each part.

Proposition 49. *Let π and π' be two while-free programs in* Prg. *If π is strongly executable on s and π' is strongly executable on t for each $t \in Q(\pi)(s)$, then any derivation of $\pi;\pi'$ is in the form of:*

$$\langle \pi;\pi', s\rangle \overset{z_1}{\Rightarrow} \langle \pi_1;\pi', s_1\rangle \overset{z_2}{\Rightarrow} \langle \pi_2;\pi', s_2\rangle \overset{z_3}{\Rightarrow} \cdots \langle \pi', t\rangle \overset{y_1}{\Rightarrow} \langle \pi'_1, t_1\rangle \overset{y_2}{\Rightarrow} \cdots t_k$$

such that $\langle \pi, s\rangle \overset{z_1}{\Rightarrow} \langle \pi_1, s_1\rangle \overset{z_2}{\Rightarrow} \cdots t$ is a derivation of π at s, and $\langle \pi', t\rangle \overset{y_1}{\Rightarrow} \langle \pi'_1, t_1\rangle \overset{y_2}{\Rightarrow} \cdots t_k$ is a derivation of π' at t.

Proof. We do induction on the complexity of π.

If $\pi = a$ or $\pi = \epsilon$, then no matter what π' is, it is straightforward to show the claim due to Atom, Skip and CompoR, and the fact that π' is strongly executable at each $t \in Q(\pi)(s)$.

Suppose $\pi = \eta;\eta'$, and $\langle(\eta;\eta');\pi', s\rangle \overset{z}{\Rightarrow} \gamma$. First note that due to CompoR and CompoL, it is not possible to have $(\eta;\eta') \overset{z}{\Rightarrow} w$ for some state w in the model. Therefore we can only use CompoL to derive $\langle(\eta;\eta');\pi', s\rangle \overset{z}{\Rightarrow} \gamma$, which means γ must be in the shape of $\langle\eta'';\pi', s'\rangle$ such that $\langle\eta;\eta', s\rangle \overset{z}{\Rightarrow} \langle\eta'', s'\rangle$. Since $\eta;\eta'$ is strongly executable at s then η'' is strongly executable at s' by the definition of strong executability. Now η'' is less complex than $\eta;\eta'$ (due to the above observation) and π' is still strongly executable at any $t \in Q(\eta'')(s') \subseteq Q(\eta;\eta')(s)$ by the definition of Q. Thus by IH, any follow-up derivation of $\langle\eta'';\pi', s'\rangle$ will be in the right shape and we can attach $\langle\eta;\eta', s\rangle \overset{z}{\Rightarrow} \langle\eta'', s'\rangle$ in front of the derivation for η'' to have the derivation for $\langle\eta;\eta', s\rangle$.

Suppose $\pi = \texttt{if } \varphi \texttt{ then } \eta \texttt{ else } \eta'$ and $\langle\pi;\pi', s\rangle \overset{z}{\Rightarrow} \gamma$. According to IfthenP and IfthenN, $\langle\pi, s\rangle \overset{\epsilon}{\Rightarrow} \langle\eta, s\rangle$ if $\mathcal{M}, s \vDash \varphi$ and $\langle\pi, s\rangle \overset{\epsilon}{\Rightarrow} \langle\eta', s\rangle$ if $\mathcal{M}, s \nvDash \varphi$. Therefore by CompoR, and CompoL, $z = \epsilon$ and γ is either $\langle\eta;\pi, s\rangle$ or $\langle\eta';\pi, s\rangle$. Note that since π is strongly executable at s then η is strongly executable at s if $\mathcal{M}, s \vDash \varphi$, and η' is strongly executable at s if $\mathcal{M}, s \nvDash \varphi$, by the definition of strong executability. We can then prove the claim like the above case using IH. □

The following is then a straightforward consequence of the Proposition 49.

Proposition 50. *Let π_1 and π_2 be two while-free programs. If π_1 is strongly executable on s and π_2 is strongly executable on t for each $t \in Q(\pi_1)(s)$, we then have that $\pi_1;\pi_2$ is strongly executable on s.*

Now we can finally show that the composition axiom is valid.

Lemma 51. *For each finite model $\mathcal{M}$, we have that $\mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h(\varphi \vee \mathcal{K}h\chi) \to \mathcal{K}h(\varphi \vee \chi)$.*

Proof. Assuming $\mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h(\varphi \vee \mathcal{K}h\chi)$, we need to show that $\mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h(\varphi \vee \chi)$. Since $\mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h(\varphi \vee \mathcal{K}h\chi)$, there is a CK-plan θ such that (1) θ is strongly executable on $[s]$; and (2) $\mathcal{M}, t \vDash_{\mathrm{CK}} \varphi \vee \mathcal{K}h\chi$ for each $t \in Q(\theta)([s])$. If $\mathcal{M}, t \vDash_{\mathrm{CK}} \mathcal{K}h\chi$, then there is a CK-plan $\theta_{[t]}$ such that (2.1) $\theta_{[t]}$ is strongly executable on $[t]$; and (2.2) $\mathcal{M}, v \vDash_{\mathrm{CK}} \chi$ for each $v \in Q(\theta_{[t]})([t])$.

Let $E = \{[t] \mid t \in Q(\theta)([s])\}$, and let D be a maximal subset of E such that $\mathcal{M}, [x] \not\leftrightarrows \mathcal{M}, [y]$ for all $[x], [y] \in D$, namely D is a set of non-bisimilar representatives in E. Since $\mathcal{M}$ is finite, D is a finite set. We enumerate all elements in D as $D = \{[t_1], \cdots, [t_n]\}$. It is obvious that for each $[t] \in E$ there is some $[t_i] \in D$ such that $\mathcal{M}, [t] \leftrightarrows \mathcal{M}, [t_i]$. By Proposition 8, for each $[t_i] \in D$ there is an epistemic formula $\mathcal{K}\chi$ in the language of EAL that is true on $[t_i]$ but false on all other elements of D. To see this, note that for each $[t_i] \in D$, $\mathcal{M}, [t_i] \not\leftrightarrows \mathcal{M}, [t_j]$ if $i \neq j$. Then by Proposition 8, there is some epistemic formula $\psi_{i,j} \in \mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ such that $\mathcal{M}, t_i \vDash \psi_{i,j}$ and $\mathcal{M}, t_j \nvDash \psi_{i,j}$.[12] Let $\psi_i = \mathcal{K} \bigwedge_{1 \leq j \leq n, j \neq i} \psi_{i,j}$. Then ψ_i is true on states in $[t_i]$ but false on states in $[t_j]$ where $i \neq j$. Thus, for each $1 \leq i \leq n$, ψ_i is the epistemic formula such that $\mathcal{M}, t_i \vDash \psi_i$ and $\mathcal{M}, t_j \nvDash \psi_i$ for all j with $i \neq j$. Further more we have:

(∗) For each $[t] \in E$, there is some i such that $\mathcal{M}, t \vDash \psi_i$, and if $\mathcal{M}, t \vDash \psi_i$ for some i then we have $\mathcal{M}, [t] \leftrightarrows \mathcal{M}, [t_i]$ by the definition of D.

For each $[t_i] \in D$, let $\theta_{[t_i]} = \epsilon$ if $\mathcal{M}, t_i \nvDash_{\mathrm{CK}} \mathcal{K}h\chi$, and otherwise let $\theta_{[t_i]}$ be a witness CK-plan satisfying (2.1) and (2.2). We then have that $\theta_{[t_i]}$ is strongly executable on $[t_i]$ for each $[t_i] \in D$.

Next, we define a series of conditional plans ξ_i where $1 \leq i \leq n$ based on $\theta_{[t_1]} \cdots \theta_{[t_n]}$ and $\psi_1 \cdots \psi_n$. ξ_i is defined, by induction on i, as follows:

$$\xi_1 := \texttt{if } \psi_1 \texttt{ then } \theta_{[t_1]} \texttt{ else } \epsilon$$
$$\xi_{i+1} := \texttt{if } \psi_{i+1} \texttt{ then } \theta_{[t_{i+1}]} \texttt{ else } \xi_i$$

It is obvious that ξ_i for all $1 \leq i \leq n$ is a CK-plan. To show that $\mathcal{M}, s \vDash_{\mathrm{CK}} \mathcal{K}h(\varphi \vee \chi)$, we only need to show that $\theta;\xi_n$ is strongly executable on $[s]$ and that $\mathcal{M}, v \vDash_{\mathrm{CK}} \varphi \vee \chi$ for each $v \in Q(\theta;\xi_n)([s])$. Next, we will prove the following two claims.

[12] Note that $\mathcal{M}, t_i \nvDash \mathcal{K}\psi$ iff $\mathcal{M}, t_i \nvDash \mathcal{K}\neg\mathcal{K}\psi$ where $\mathcal{K}\neg\mathcal{K}\psi$ is an epistemic formula in EAL.

Claim 51.1. *The plan $\theta; \xi_n$ is strongly executable on $[s]$.*

Proof of Claim 51.1. By Proposition 50, we only need to show that θ is strongly executable on $[s]$ and that ξ_n is strongly executable on $[t]$ for each $t \in Q(\theta)([s])$. By (1), we have that θ is strongly executable on $[s]$. We then only need to show that ξ_n is strongly executable on $[t]$ for each $t \in Q(\theta)([s])$. In other words, we only need to show that ξ_n is strongly executable on each $[t] \in E$. We will prove a stronger proposition:

$$\text{for each } 1 \leq i \leq n,\ \xi_i \text{ is strongly executable on each } [t] \in E.$$

We prove it by induction on i. For the case of ξ_1, for each $[t] \in E$, either $\mathcal{M}, t \vDash \psi_1$ or $\mathcal{M}, t \nvDash \psi_1$.[13] $\theta_{[t_1]}$ is clearly strongly executable, if $\mathcal{M}, t \nvDash \psi_1$. If $\mathcal{M}, t \vDash \psi_1$, it follows by $(*)$ that $\mathcal{M}, [t] \leftrightarrows \mathcal{M}, [t_1]$. Please note that $\theta_{[t_1]}$ is strongly executable on $[t_1]$. Thus, it follows by Proposition 26 that $\theta_{[t_1]}$ is strongly executable on $[t]$. In sum, ξ_1 is strongly executable on each $[t] \in E$. For the case of ξ_{i+1}, it is straightforward by IH, if $\mathcal{M}, t \nvDash \psi_{i+1}$. If $\mathcal{M}, t \vDash \psi_{i+1}$, for a similar reason as in the case of ξ_1 based on the IH, we also have that $\theta_{[t_{i+1}]}$ is strongly executable on $[t]$. Thus, ξ_{i+1} is strongly executable on $[t]$. ■

Claim 51.2. *$\mathcal{M}, v \vDash_{\mathrm{CK}} \varphi \vee \chi$ for each $v \in Q(\theta; \xi_n)([s])$.*

Proof of Claim 51.2. Given $v \in Q(\theta; \xi_n)([s])$, it follows that there is some $t \in Q(\theta)([s])$ such that $v \in Q(\xi_n)(t)$. Clearly $[t] \in E$. By the definition of the set D, we know that there is some $[t_i] \in D$ such that $\mathcal{M}, [t] \leftrightarrows \mathcal{M}, [t_i]$. By Proposition 8 and the feature of ψ_i, we then have that $\mathcal{M}, t \vDash \psi_i$ and that $\mathcal{M}, t \nvDash \psi_j$ for each $1 \leq j \leq n$ with $i \neq j$. By the definition of ξ_n and $v \in Q(\xi_n)(t)$, we have that $v \in Q(\theta_{[t_i]})(t)$.

If $\mathcal{M}, t_i \nvDash_{\mathrm{CK}} \mathcal{K}h\chi$, by definition $\theta_{[t_i]} = \epsilon$. Thus, we have $v = t$, and we need to show that $\mathcal{M}, t \vDash_{\mathrm{CK}} \varphi \vee \chi$. Since $\mathcal{M}, [t] \leftrightarrows \mathcal{M}, [t_i]$, by definition $\mathcal{M}, t \leftrightarrows \mathcal{M}, t'$ for some $t' \in [t_i]$. Moreover, $\mathcal{M}, t' \nvDash_{\mathrm{CK}} \mathcal{K}h\chi$ because of $t' \in [t_i]$. It follows by Theorem 28 that $\mathcal{M}, t \nvDash_{\mathrm{CK}} \mathcal{K}h\chi$. Since $t \in Q(\theta)([s])$, $\mathcal{M}, t \vDash_{\mathrm{CK}} \varphi \vee \mathcal{K}h\chi$, thus we have $\mathcal{M}, t \vDash_{\mathrm{CK}} \varphi$.

If $\mathcal{M}, t_i \vDash_{\mathrm{CK}} \mathcal{K}h\chi$, since $\mathcal{M}, [t] \leftrightarrows \mathcal{M}, [t_i]$ and $v \in Q(\theta_{[t_i]})(t)$, it follows by Proposition 27 that there is some $v' \in Q(\theta_{[t_i]})([t_i])$ such that $\mathcal{M}, v \leftrightarrows \mathcal{M}, v'$. Since $\mathcal{M}, t_i \vDash_{\mathrm{CK}} \mathcal{K}h\chi$ and $v' \in Q(\theta_{[t_i]})([t_i])$, $\mathcal{M}, v' \vDash_{\mathrm{CK}} \chi$ because of (2.2). Thus, it follows by Theorem 28 that $\mathcal{M}, v \vDash_{\mathrm{CK}} \chi$. ■ □

Based on the above lemma, the soundness follows.

Theorem 52 *(Soundness). The proof system $\mathbb{SLKHC}$ is sound with respect to $\vDash_{\mathrm{CK}}$ over finite models.*

5.2. Completeness

In this section, we construct a finite canonical model of $\mathbb{SLKHC}$ for the $\vDash_{\mathrm{CK}}$ semantics. Let Φ be a subformula-closed set of $\mathcal{L}^{\mathsf{ELKh}}$-formulas that is finite. Recall that $cl(\Phi)$ is the closure $\Phi \cup \{\mathcal{K}\varphi, \neg\mathcal{K}\varphi \mid \varphi \in \Phi\}$. Since Φ is finite, so is $cl(\Phi)$. Next, we will use it to build a canonical model with respect to Φ.

Definition 53 *(Atom).* We enumerate the formulas in $cl(\Phi)$ by $\{\psi_0, \cdots, \psi_h\}$ where $h \in \mathbb{N}$. The formula set $\Delta = \{Y_i \mid i \leq h\}$ is an atom of $cl(\Phi)$ if

- $Y_i = \psi_i$ or $Y_i = \neg\psi_i$ for each $\psi_i \in cl(\Phi)$;
- Δ is consistent in $\mathbb{SLKHC}$.

By a standard inductive construction, we can obtain the Lindenbaum-like result in our setting (which is useful to show the existence lemma for $\mathcal{K}$):

Proposition 54. *Let Δ be an atom of $cl(\Phi)$, $\Gamma \subseteq \Delta$ and $\varphi \in cl(\Phi)$. If $\Gamma \cup \{\pm\varphi\}$ is consistent then there is an atom Δ' of $cl(\Phi)$ such that $(\Gamma \cup \{\pm\varphi\}) \subseteq \Delta'$, where $\pm\varphi = \varphi$ or $\pm\varphi = \neg\varphi$.*

Proposition 55. *Let Δ be an atom of Φ, and $\mathcal{K}\varphi \in cl(\Phi)$. If $\mathcal{K}\varphi \notin \Delta$ then there exists Δ' such that $\Delta'|_{\mathcal{K}} = \Delta|_{\mathcal{K}}$ and $\neg\varphi \in \Delta'$.*

Proof. Since $\mathcal{K}\varphi \in cl(\Phi)$, by the definition of $cl(\Phi)$, we have $\varphi \in \Phi$. Let $\Gamma = (\Delta|_{\mathcal{K}} \cup \Delta|_{\neg\mathcal{K}})$. We then have that $\Gamma \subseteq \Delta$. Firstly, we show that $\Gamma \cup \{\neg\varphi\}$ is consistent. If it is not consistent, then there are $\mathcal{K}\psi_1 \cdots \mathcal{K}\psi_n \in \Delta|_{\mathcal{K}}$, and $\neg\mathcal{K}\chi_1 \cdots \neg\mathcal{K}\chi_m \in \Delta|_{\neg\mathcal{K}}$ such that $\vdash \mathcal{K}\psi_1 \wedge \cdots \wedge \mathcal{K}\psi_n \wedge \neg\mathcal{K}\chi_1 \wedge \cdots \wedge \neg\mathcal{K}\chi_m \rightarrow \varphi$. By the Rule NECK and axiom DISTK, we have that $\vdash \mathcal{K}\mathcal{K}\psi_1 \wedge \cdots \wedge \mathcal{K}\mathcal{K}\psi_n \wedge \mathcal{K}\neg\mathcal{K}\chi_1 \wedge \cdots \wedge \mathcal{K}\neg\mathcal{K}\chi_m) \rightarrow \mathcal{K}\varphi$. Since $\mathcal{K}\psi_1 \cdots \mathcal{K}\psi_n, \neg\mathcal{K}\chi_1 \cdots \neg\mathcal{K}\chi_m \in \Delta$, by the Axioms T and 4 and that Δ is maximal consistent, we have that $\mathcal{K}\mathcal{K}\psi_1 \cdots \mathcal{K}\mathcal{K}\psi_n, \mathcal{K}\neg\mathcal{K}\chi_1 \cdots \mathcal{K}\neg\mathcal{K}\chi_m \in \Delta$. From this follows that $\mathcal{K}\varphi \in \Delta$. Contradiction!

[13] Since ψ_i is an epistemic formula, the representative of $[t]$ is not essential.

Thus, we have that $\Gamma \cup \{\neg\varphi\}$ is consistent. By Proposition 54, we have that there is an atom Δ' such that $(\Gamma \cup \{\neg\varphi\}) \subseteq \Delta'$. Thus, we have that $\Delta|_{\mathcal{K}} \subseteq \Delta'$ and $\neg\varphi \in \Delta'$. Next, we only need to show that $\Delta'|_{\mathcal{K}} \subseteq \Delta|_{\mathcal{K}}$. Assume that there is $\mathcal{K}\psi \in \Delta'$ but $\mathcal{K}\psi \notin \Delta$. Since $\mathcal{K}\psi \notin \Delta$, $\neg\mathcal{K}\psi \in \Delta$. Since $\Delta|_{\neg\mathcal{K}} \subseteq \Delta'$, we then have that $\neg\mathcal{K}\psi \in \Delta'$. This is contradictory to $\mathcal{K}\psi \in \Delta$ and that Δ is consistent. Therefore, we have that $\Delta'|_{\mathcal{K}} \subseteq \Delta|_{\mathcal{K}}$. □

Proposition 56. *Let Δ and Δ' be two atoms of Φ such that $\Delta|_{\mathcal{K}} = \Delta'|_{\mathcal{K}}$. We have $\Delta|_{\mathcal{K}h} = \Delta'|_{\mathcal{K}h}$.*

Proof. For each $\mathcal{K}h\varphi \in \Delta$, by Definition 42, $\mathcal{K}h\varphi \in \Phi$. Then $\mathcal{K}\mathcal{K}h\varphi \in cl(\Phi)$. For each $\mathcal{K}h\varphi \in \Delta$, by the Axiom AxKhtoKKh, we have $\mathcal{K}\mathcal{K}h\varphi \in \Delta$. Since $\Delta|_{\mathcal{K}} = \Delta'|_{\mathcal{K}}$, then $\mathcal{K}\mathcal{K}h\varphi \in \Delta'$, and by Axiom T, $\mathcal{K}h\varphi \in \Delta'$. Then we showed that $\mathcal{K}h\varphi \in \Delta$ implies $\mathcal{K}h\varphi \in \Delta'$. Similarly we can prove $\mathcal{K}h\varphi \in \Delta'$ implies $\mathcal{K}h\varphi \in \Delta$. Hence, $\Delta|_{\mathcal{K}h} = \Delta'|_{\mathcal{K}h}$. □

Definition 57 *(Canonical model).* Given a subformula-closed Φ, the canonical model $\mathcal{M}^{\Phi} = \langle W, \mathbf{A}^{\Phi} \sim, \{Q(\varphi) \mid \varphi \in \mathbf{A}^{\Phi}\}, V\rangle$ is defined as:

- $W = \{\Delta \mid \Delta \text{ is an atom of } cl(\Phi)\}$;
- $\mathbf{A}^{\Phi} = \{\varphi \mid \mathcal{K}h\varphi \in \Phi\}$;
- for each $\varphi \in \mathbf{A}^{\Phi}$, we have that $(\Delta, \Gamma) \in Q(\varphi) \iff \{\mathcal{K}h\varphi, \neg\mathcal{K}\varphi\} \subseteq \Delta$, and $\varphi \in \Gamma$;
- $\Delta \sim \Gamma \iff \Delta|_{\mathcal{K}} = \Gamma|_{\mathcal{K}}$;
- for each $p \in \Phi$, $p \in V(\Delta) \iff p \in \Delta$.

Proposition 58. *Let Δ be a state in $\mathcal{M}^{\Phi}$ and $\psi \in \mathbf{A}^{\Phi}$ be an atomic action that is executable on Δ. If $\varphi_1 \in \Delta'$ or $\varphi_2 \in \Delta'$ for all $\Delta' \in Q(\psi)(\Delta)$ then $\Delta \vdash \mathcal{K}h(\varphi_1 \vee \varphi_2)$.*

Proof. Since $\psi \in \mathbf{A}^{\Phi}$ is executable on Δ, we then know that $\neg\mathcal{K}\psi, \mathcal{K}h\psi \in \Delta$. First, we show that ψ is not consistent with $\{\neg\varphi_1, \neg\varphi_2\}$. Assuming that ψ is consistent with $\{\neg\varphi_1, \neg\varphi_2\}$, by Proposition 54 there exists an atom Γ of $cl(\Phi)$ such that $\{\psi, \neg\varphi_1, \neg\varphi_2\} \subseteq \Gamma$. By Definition 57, we then have $\Delta \xrightarrow{\psi} \Gamma$. This is contradictory to the facts that $\varphi_1 \in \Delta'$ or $\varphi_2 \in \Delta'$ for all $\Delta' \in Q(\psi)(\Delta)$ and that Γ is consistent. Therefore, ψ is not consistent with $\{\neg\varphi_1, \neg\varphi_2\}$. It follows that $\vdash \psi \to (\varphi_1 \vee \varphi_2)$. Then, it follows by the rule MONOKh that $\vdash \mathcal{K}h\psi \to \mathcal{K}h(\varphi_1 \vee \psi_2)$. Since $\mathcal{K}h\psi \in \Delta$, $\Delta \vdash \mathcal{K}h(\varphi_1 \vee \varphi_2)$. □

Now we are ready to obtain the truth lemma.

Lemma 59. *For each $\varphi \in cl(\Phi)$, $\mathcal{M}^{\Phi}, \Delta \vDash_{\mathtt{CK}} \varphi$ iff $\varphi \in \Delta$.*

Proof. We prove it by induction on φ. We only focus on the case of $\mathcal{K}h\varphi \in cl(\Phi)$; the other cases are straightforward, e.g., $\mathcal{K}\varphi$ case can be proved based on Proposition 55. Note that if $\mathcal{K}h\varphi \in cl(\Phi)$ then $\mathcal{K}h\varphi \in \Phi$, therefore $\varphi \in \Phi$ since Φ is subformula-closed. Thus by Definition 42, $\mathcal{K}\varphi \in cl(\Phi)$.

Right to Left: If $\mathcal{K}h\varphi \in \Delta$, we will show $\mathcal{M}^{\Phi}, \Delta \vDash_{\mathtt{CK}} \mathcal{K}h\varphi$. Firstly, there are two cases: $\mathcal{K}\varphi \in \Delta$ or $\mathcal{K}\varphi \notin \Delta$.

If $\mathcal{K}\varphi \in \Delta$, then $\mathcal{K}\varphi, \varphi \in \Delta'$ for all $\Delta' \in [\Delta]$. Since $\varphi \in \Phi$, it follows by IH that $\mathcal{M}^{\Phi}, \Delta' \vDash_{\mathtt{CK}} \varphi$ for all $\Delta' \in [\Delta]$. Therefore, $\mathcal{M}^{\Phi}, \Delta \vDash_{\mathtt{CK}} \mathcal{K}\varphi$. It follows by the Axiom AxKtoKh and the soundness of $\mathbb{SLKHC}$ that $\mathcal{M}^{\Phi}, \Delta \vDash_{\mathtt{CK}} \mathcal{K}h\varphi$.

If $\neg\mathcal{K}\varphi \in \Delta$, we first show that φ is consistent. If not, namely $\vdash \varphi \to \bot$, it follows by the rule MONOKh that $\vdash \mathcal{K}h\varphi \to \mathcal{K}h\bot$. It follows by the Axiom AxKhbot that $\vdash \mathcal{K}h\varphi \to \bot$. Since $\mathcal{K}h\varphi \in \Delta$, this is in contradiction with the fact that Δ is consistent. Therefore, φ is consistent. By Proposition 54 there exists an atom Δ' such that $\varphi \in \Delta'$. Since $\mathcal{K}h\varphi \in \Delta$, $\varphi \in \mathbf{A}^{\Phi}$. Next, we will show that $\varphi \in \mathbf{A}^{\Phi}$ is a proper CK-plan for $\mathcal{M}^{\Phi}, \Delta \vDash_{\mathtt{CK}} \mathcal{K}h\varphi$.

Note that $\{\mathcal{K}h\varphi, \neg\mathcal{K}\varphi\} \subseteq \Delta$ implies for all $\Gamma \in [\Delta]$, $\{\mathcal{K}h\varphi, \neg\mathcal{K}\varphi\} \subseteq \Gamma$ by Proposition 56 and the definition of the canonical epistemic relation $\sim$. Now since $\varphi \in \Delta'$, by Definition 57 we have that $\varphi \in \mathbf{A}^{\Phi}$ is executable at each $\Gamma \in [\Delta]$, because $\Gamma \xrightarrow{\varphi} \Delta'$ for each $\Gamma \in [\Delta]$. What is more, for each $\Gamma' \in Q(\varphi)([\Delta])$, we then have $\varphi \in \Gamma'$ by Definition 57. By IH, it follows that $\mathcal{M}^{\Phi}, \Gamma' \vDash_{\mathtt{CK}} \varphi$ for each $\Gamma' \in Q(\varphi)([\Delta])$. Therefore, we have $\mathcal{M}^{\Phi}, \Delta \vDash_{\mathtt{CK}} \mathcal{K}h\varphi$.

Left to Right: Suppose $\mathcal{M}^{\Phi}, \Delta \vDash_{\mathtt{CK}} \mathcal{K}h\varphi$, we will show $\mathcal{K}h\varphi \in \Delta$. By the semantics, there exists a CK-plan θ such that θ is strongly executable on $[\Delta]$ and $\mathcal{M}^{\Phi}, \Delta' \vDash_{\mathtt{CK}} \varphi$ for each $\Delta' \in Q(\theta)([\Delta])$. We will show that $\mathcal{K}h\varphi \in \Delta$ holds in both the two situations: all complete traces of θ on all $\Gamma \in [\Delta]$ are of length 1; or there is some complete trace τ of θ on some $\Gamma \in [\Delta]$ such that the length of τ is bigger than 1.

- If all complete traces of θ on all $\Gamma \in [\Delta]$ are of length 1, then it means that $Q(\theta)([\Delta]) = [\Delta]$. Since $\mathcal{M}^{\Phi}, \Delta' \vDash_{\mathtt{CK}} \varphi$ for each $\Delta' \in Q(\theta)([\Delta])$, we then have that $\varphi \in \Delta'$ for each $\Delta' \in [\Delta]$ by IH. Moreover, since $\mathcal{K}h\varphi \in cl(\Phi)$, we then have $\mathcal{K}\varphi \in cl(\Phi)$ by Definition 42. Since $\varphi \in \Delta'$ for each $\Delta' \in [\Delta]$, it follows by Proposition 55 that $\mathcal{K}\varphi \in \Delta$. By AxKtoKh, we then have that $\mathcal{K}h\varphi \in \Delta$ since Δ is consistent.
- Next, we will show $\mathcal{K}h\varphi \in \Delta$ also holds if there is some complete trace τ of θ on some $\Gamma \in [\Delta]$ such that the length of τ is bigger than 1. Firstly, we show the following claim:

Claim 59.1. *If $\Gamma_0 \cdots \Gamma_n$ is a non-complete trace of θ on some $\Gamma \in [\Delta]$ and $\neg \mathcal{K}h\varphi \in \Gamma_n$, then for each $i \in \mathbb{N}$ there is a non-complete trace $\Gamma_0 \cdots \Gamma_n \cdots \Gamma_{n+i}$ such that $\neg \mathcal{K}h\varphi \in \Gamma_{n+i}$.*

Proof of Claim 59.1. We prove it by induction on i. It is obvious if $i = 0$. With the IH that the claim holds for i, we will prove it also holds for $i+1$. Let $\Gamma_0 \cdots \Gamma_{n+i}$ be a non-complete trace of θ on $\Gamma \in [\Delta]$ and $\neg \mathcal{K}h\varphi \in \Gamma_{n+i}$. It follows that there are some $\chi \in \mathbf{A}^{\Phi}$ and some $\Gamma' \in Q(\chi)(\Gamma_{n+i})$ such that $\Gamma_0 \cdots \Gamma_{n+i} \chi \Gamma'$ is a partial trace of θ on Γ. What is more, since θ is strongly executable on $[\Gamma]$, by Proposition 17, we have that $\Gamma_0 \cdots \Gamma_{n+i} \chi \Gamma'$ is a partial trace of θ on Γ for each $\Gamma' \in Q(\chi)(\Gamma_{n+i})$.
If all partial traces $\Gamma_0 \cdots \Gamma_{n+i} \chi \Gamma'$ where $\Gamma' \in Q(\chi)(\Gamma_{n+i})$ are complete traces, then $Q(\chi)(\Gamma_{n+i}) \subseteq Q(\theta)([\Delta])$. Thus, we have $\varphi \in \Gamma'$ for each $\Gamma' \in Q(\chi)(\Gamma_{n+i})$. By (a special case of) Proposition 58, it follows that $\mathcal{K}h\varphi \in \Gamma_{n+i}$. This is contradictory to $\neg \mathcal{K}h\varphi \in \Gamma_{n+i}$. Therefore, there exists a partial trace $\Gamma_0 \cdots \Gamma_{n+i} \chi \Gamma'$ for some $\Gamma' \in Q(\chi)(\Gamma_{n+i})$ that is not complete.
Assume towards contradiction that for each non-complete trace $\Gamma_0 \cdots \Gamma_{n+i} \chi \Gamma'$ where $\Gamma' \in Q(\chi)(\Gamma_{n+i})$, we have $\mathcal{K}h\varphi \in \Gamma'$. Note that if $\Gamma_0 \cdots \Gamma_{n+i} \chi \Gamma'$ is complete then $\varphi \in \Gamma'$ since $\mathcal{M}^{\Phi}, \Delta \vDash_{\mathrm{CK}} \mathcal{K}h\varphi$. Combining the above two cases of complete and non-complete traces, we have that $\varphi \in \Gamma'$ or $\mathcal{K}h\varphi \in \Gamma'$ for each $\Gamma' \in Q(\chi)(\Gamma_{n+i})$. It follows by Proposition 58 that $\Gamma_{n+i} \vdash \mathcal{K}h(\varphi \vee \mathcal{K}h\varphi)$. By the Axiom AxKhKhG, it follows that $\Gamma_{n+i} \vdash \mathcal{K}h\varphi$, which is contradictory to $\neg \mathcal{K}h\varphi \in \Gamma_{n+i}$. Therefore, there is a non-complete execution trace of $\Gamma_0 \cdots \Gamma_{n+i+1}$ such that $\neg \mathcal{K}h\varphi \in \Gamma_{n+i+1}$. ■

If there is some complete trace τ of θ on some $\Gamma \in [\Delta]$ such that the length of τ is bigger than 1, this means that there is a complete trace $\Gamma_0 \cdots \Gamma_n$ of θ on some $\Gamma \in [\Delta]$, where $n \geq 1$. Then, we have that $\mathcal{K}h\varphi \in \Gamma_i$ for all $0 \leq i < n$. Otherwise, by Claim 59.1, there can be arbitrarily long partial trace of θ on Γ. This is contradictory to the fact that the length of the complete trace of θ is bounded by the length of θ since θ is while-free. Thus, we have $\mathcal{K}h\varphi \in \Gamma$. Since $\Gamma \in [\Delta]$, by Proposition 56, we have that $\mathcal{K}h\varphi \in \Delta$. □

Theorem 60. *The proof system $\mathbb{SLKHC}$ is weakly complete with respect to $\vDash_{\mathrm{CK}}$ over finite models.*

By Proposition 48, we have:

Corollary 61. *The proof system $\mathbb{SLKHC}$ is sound and weakly complete with respect to $\vDash_{\mathrm{K}}$ over finite models.*

The following example demonstrates that $\mathbb{SLKHC}$ cannot be strongly complete over finite models.

Example 62. Let $\Gamma = \{\mathcal{K}hp_1, \mathcal{K}h(\neg p_1 \wedge p_2), \mathcal{K}h(\neg p_1 \wedge \neg p_2 \wedge p_3), \cdots\}$. Γ is consistent but not satisfiable with respect to $\vDash_{\mathrm{K}}$ or $\vDash_{\mathrm{CK}}$ on finite models since it requires an infinite model.

By the construction of the finite canonical model, we know that the logic has a small model property with respect to $\vDash_{\mathrm{CK}}$. The decidability then follows:

Theorem 63 *(Decidability). The logic $\mathbb{SLKHC}$ is decidable.*

6. Discussions

In this section, we discuss several choices that we made in the paper and the relevance of our work to epistemic planning:

- Section 6.1 discusses the choice of the logical language and the point of staying at a suitable abstraction level to reveal the general principles of epistemic planning.
- Section 6.2 discusses the choice of the plan specification language and why syntactic specification of plans matters for epistemic planning.
- Section 6.3 discusses the extensions to the multi-agent setting that can be and cannot be captured by the current framework.
- Section 6.4 discusses two extra properties about the interactions of knowledge and actions which are often implicitly assumed in the literature of epistemic planning.
- Section 6.5 discusses the choice of using models with explicit states, and the technical connection to compact representations often used in the literature.
- Section 6.6 discusses how to view generalized planning problems as model checking problems in our framework and prove the PTIME complexity in one case.
- Section 6.7 discusses model theoretical and proof theoretical questions about epistemic planning arising naturally in our framework.

6.1. On the choice of the logical language

In this paper, we introduced three formal languages: the logical language $\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$, the plan specification language $\mathtt{Prg}$, and the logical language of know-how $\mathcal{L}^{\mathsf{ELKh}}$. $\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ is only used to define the conditions in the conditional plans. Fragments of $\mathtt{Prg}$ are used to define various notions of plans, where the conditional ones use the formulas in $\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$. $\mathcal{L}^{\mathsf{ELKh}}$, the epistemic language of know-how, is our working language for the axiomatization, whose semantics uses $\mathtt{Prg}$ (thus also $\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$) to define the notions of plans. Note that the action modalities $[a]$ of $\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$ do not appear in $\mathcal{L}^{\mathsf{ELKh}}$, and the $\mathcal{K}h$ modality of $\mathcal{L}^{\mathsf{ELKh}}$ does not appear in $\mathcal{L}_{\mathbf{A}}^{\mathsf{EAL}}$.[14] The language $\mathcal{L}^{\mathsf{ELKh}}$ serves the main aim of the paper to obtain the general principles behind know-how, as explained below.

Theorem 40 shows that the proof system $\mathbb{SLKH}$ is sound and complete with respect to *all* the ten semantics of $\mathcal{K}h$ that we have introduced in the paper. The result is actually even more general. It shows that the completeness holds for all the similar semantics for $\mathcal{K}h$ in the generic form of Definition 20 satisfying the two properties in Proposition 23. One may then argue that this very result also shows that our logical language $\mathcal{L}^{\mathsf{ELKh}}$ is *too weak* to distinguish all these different notions of know-how. However, this is not the case. Firstly, at the model-level, different semantics indeed induce different notions of know-how, as shown by Proposition 22. On the other hand, at the frame-level, we do identify the shared general principles of know-how no matter what specific notion of the plan is used. Further differences of various notions of know-how can be identified by extra axioms corresponding to extra constraints over frames, such as the compositionality axiom for conditional plans over finite models, as we discussed in Section 5 (see also Section 6.4 for related discussions).

The main purpose of this paper, as the title shows, is exactly to find *the* logic of knowing how unifying the previous approaches in terms of general principles shared by various notions. In this way, we also hope to reveal the underlying core principles of epistemic planning independent of the specific plan in use. It is common in philosophical logic to design a suitable language as the "looking glasses" to reveal the general properties that we care at a particular abstraction level. We would not be able to discover such principles about know-how if we allow a much stronger language that breaks the $\mathcal{K}h$ modality into detailed components such as PDL-like modalities, quantifiers and so on. This is also in line with the general research program of the logics of know-wh proposed by [52]. An axiomatization using such a more expressive language will not directly tell us about the properties of know-how, but instead, we will be drowned in the detailed properties about quantifiers and program constructors as in the case of dynamic logic.

Having argued for using a weak language for our purpose in this paper, we are nevertheless not against moving to a more expressive language if further distinctions between different plans are needed. Actually, it is exciting to see how it may differ from the usual PDL setting if we include $[\pi]$ modalities in the language and reinterpret $[\pi]\varphi$ as π is *strongly executable* and it can guarantee φ eventually.

6.2. On the choice of the plan specification language

In AI planning, the mathematical notion of a plan is often dependent on the given model, e.g., a (partial) function assigning actions to epistemic equivalence classes as in contingent planning. However, as in the case of programs and protocols, the plans are better understood as syntactic objects independent from the models. Not only because it is very natural to write down a plan like a program, especially when it involves non-trivial conditions and (nested) loops, but also because we can discuss plans as stand-alone objects. For example, in this way, we can meaningfully say sometimes the *same* plan works for *different* models (for the same goal), though it may generate quite different execution paths on those models. We can see why one plan works on this model but not that model, and we can improve the plan or combine the plans syntactically. The literature of theoretical computer science, such as the work on process algebra, provides us many technical tools to discuss the equivalence of plans or compare them without fixing the models (cf. e.g., [7]).

For such a syntactic approach, a plan specification language and its corresponding formal semantics are needed. Various existing notions of plans can then be considered as fragments of the plan specification language such that we can compare them formally. Our choice of the specification language is inspired by the knowledge-based program approach to planning [22,23]. Unlike the sketch in [23], we rigorously provide a non-trivial structural operational semantics of this specification language in order to define the execution traces of a plan to be used in the semantics of $\mathcal{K}h$. The plan specification language deserves further investigations on its own. For example, we may discuss the equivalence of plans without fixing a model, and the relative expressive power of planning in terms of know-how, demonstrated by our preliminary result (Proposition 22).

6.3. On single-agent vs. multi-agent

In this work, we focus on the single-agent case to make the technical presentation less heavy. In contrast to the framework of [51], there is no essential technical difficulty to generalize the current framework to the multi-agent case: we can simply add different epistemic relations for different agents to interpret the agent-indexed modalities $\mathcal{K}_i$ and $\mathcal{K}h_i$. The proof systems can be turned into the multi-agent version by adding the indexes, and the main completeness proof may

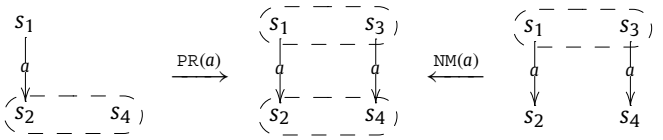
[14] Allowing $\mathcal{K}h$ in the conditional plans may lead to some circularity in the semantics for $\mathcal{K}h$.

work in principle. This will capture non-trivial *single-agent planning* in a *multi-agent setting* (note that the $\mathcal{K}h_i$ modality is still about a single agent). For example, we can express $\mathcal{K}h_i(p \wedge \neg\mathcal{K}h_j\neg p)$: agent i knows how to make sure not only p but also j does not know how to turn off p from then on. In a more cooperative scenario, we can for example express $\neg\mathcal{K}h_i p \wedge \mathcal{K}_i\neg\mathcal{K}h_j p \wedge \mathcal{K}h_i\mathcal{K}h_j p$ which captures some kind of high-level planning using others' abilities. The language of EAL can also be extended to a multi-agent setting such that we can have nested epistemic conditions in plans, such as $\mathcal{K}_i\neg\mathcal{K}_j p$. We can make it more interesting by assigning different agents with different available actions as in [31,57]. Note that adding multiple agents may affect the computational aspects of the logic and its model checking problem.

To handle multi-agent know-how or planning where joint actions play a role, we need to introduce group know-how as in [39]. On the other hand, the current multi-agent know-how approach, such as [39], can only handle one-step plans (like our simple plans) due to the technical (and conceptual) problem of synchronizing various steps of the joint plan. We leave this as an interesting problem for future research.

6.4. On extra properties about knowledge and actions

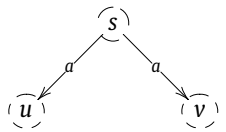
In this paper, to stay as general as possible, we do not assume any interaction properties about knowledge and actions in the models. However, there are natural properties that involve the interaction between the epistemic relation and the actions, such as (a version of) *perfect recall* and *no miracles* (cf. e.g. [54,56]). Perfect recall (PR) and no miracles (NM) correspond to the following properties depicted below:



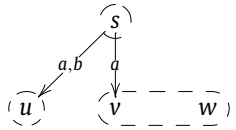
PR(a) requires that if $s_1 \xrightarrow{a} s_2$ and $s_2 \sim s_4$ then there must exist a state s_3 such that $s_1 \sim s_3$ and $s_3 \xrightarrow{a} s_4$. In words, if the agent cannot distinguish two states after doing a then it could not distinguish them before. NM(a) requires that if $s_1 \sim s_3$, $s_3 \xrightarrow{a} s_4$, and $s_1 \xrightarrow{a} s_2$ then we have that $s_2 \sim s_4$. In words, if the agent cannot distinguish two states and exactly the same thing happens on both, then it cannot distinguish the resulting two states. Prefect recall and no miracles are often assumed implicitly in dynamic epistemic logic [54,53]. Perfect recall is often also behind the definition of contingent planning, as we will show in Section 6.5.

These two properties can be easily defined by EAL formulas $\mathcal{K}[a]p \to [a]\mathcal{K}p$ and $\langle a\rangle\mathcal{K}p \to \mathcal{K}[a]p$, respectively. However, our language of knowing how does not include the action modality $[a]$ in EAL.

Nevertheless, these properties have an impact on the logic of knowing how. Take, for example, the know-how logic based on linear plans. Under the semantics of $\models_{\mathrm{L}}$, the axiom $\mathcal{K}h\mathcal{K}h\varphi \to \mathcal{K}h\varphi$ is valid over models with *no miracles*, and the axiom $\mathcal{K}h\varphi \to \mathcal{K}h\mathcal{K}\varphi$ (cf. [16]) is valid over models with *perfect recall*. However, these axioms do not characterize the corresponding properties. Consider the model $\mathcal{M}$ depicted below:



It is not hard to check that the axiom $\mathcal{K}h\mathcal{K}h\varphi \to \mathcal{K}h\varphi$ is valid under $\models_{\mathrm{L}}$ over the model $\mathcal{M}$, but it is evident that $\mathcal{M}$ does not have the property of no miracles. For the axiom $\mathcal{K}h\varphi \to \mathcal{K}h\mathcal{K}\varphi$ and the property of perfect recall, consider the model $\mathcal{M}'$ depicted below:



It is obvious that $\mathcal{M}'$ does not have the property of perfect recall, but it can be checked that $\mathcal{K}h\varphi \to \mathcal{K}h\mathcal{K}\varphi$ is valid under $\models_{\mathrm{L}}$ over the model $\mathcal{M}'$. We leave the detailed discussion on such properties for future work.

6.5. On the choice of explicit state-based models

Our approach in this article is based on models with explicit states (epistemic transition systems), but not on the compact representations of the states and actions often seen in practical planning approaches. We made this deliberative choice because we focus on the *theoretical foundation* unifying various notions of planning and know-how in this paper. The explicit models abstract away the differences of the representations in various applications and their implicit assumptions (see below for an example). This choice is also consciously made in the classic textbooks of automated planning such as [19,18]: when

introducing various planning problems in a unified framework, the state-based models with various restrictions on them *are* chosen as the underlying conceptual models.

Nevertheless, we can well connect the state-based models with compact representations, as we already did in our previous work such as [32]. Given a state variable representation where actions are represented with pre- and post-conditions, the corresponding explicit epistemic transition system can be generated by taking all valuations of the state variables as the explicit states, and computing the transitions among the valuations according to the pre- and post-conditions of the actions.

However, not all the epistemic transition systems can be generated in this way, as discussed in [32]. States with the same valuation may differ in their available actions or higher-order epistemic structure (especially in a multi-agent setting). The exact formulation of the pre- and post-conditions and the progression of the actions often implicitly assume some properties of the generated transitions. To make all these discussions more concrete, let us take the compact representation used in [23] as an example.

A *knowledge-based planning problem* (a compact model with a goal) in [23] is a quadruple $P = (I, A_O, A_E, \varphi)$ where I is a set of valuations from $\mathbf{P}$ to $\{0, 1\}$ capturing the uncertainty about the initial states, A_O is a set of *ontic actions* that change the basic facts, A_E is a set of *epistemic actions* that only change the knowledge of the agent, and the goal formula φ is an epistemic formula expressible in a modal language with the know-that operator $\mathcal{K}$ only. Each ontic action $o \in A_O$ is represented by a propositional formula which essentially tells how the truth values of propositional variables are changed by the action. For example, $p' \leftrightarrow \neg p$ means that the variable p will have the opposite value after the action, where p' denotes the truth value of p in the resulting state. An epistemic action is represented by a list of positive epistemic formulas $\langle \mathcal{K}\varphi_1, \dots \mathcal{K}\varphi_n \rangle$ (φ_i are propositional), which captures the exhaustive possible epistemic effects (not necessarily mutually exclusive). For example, $\langle \mathcal{K}p, \mathcal{K}\neg p \rangle$ captures the action of testing p: after the action either you know p or you know $\neg p$ (see [23] for the precise definitions).

Technically, the states are those sets in $2^{\mathbf{P}}$ (the *valuations*), the uncertainty over the states can be captured by the *sets of states* in $2^{2^{\mathbf{P}}}$ (also called *knowledge states*), the ontic actions can be viewed as a binary relation R_o over $2^{\mathbf{P}}$ in the most intuitive way, and each epistemic action $e \in A_E$ can be viewed as a binary relation R_e over $2^{2^{\mathbf{P}}}$ such that BR_eB' iff $B' = \{s \in B \mid s \vDash \varphi_i\} \neq \emptyset$ for some $\mathcal{K}\varphi_i$ in e. It follows that BR_eB' implies that $B' \subseteq B$. Intuitively, an epistemic action eliminates some uncertainty and obtain more knowledge.

Given R_o and R_e, we can now transform a compact representation $C = (I, A_O, A_E)$ into an epistemic transition system $\mathcal{M}^C = \langle W, \mathbf{A}, \sim, \{Q(a) \mid a \in \mathbf{A}\}, V \rangle$ as follows:

- $W = \{(B, s) \mid B \in 2^{2^{\mathbf{P}}}, s \in B\}$;
- $\mathbf{A} = A_O \cup A_E$;
- $(B, s) \sim (B', s') \iff B = B'$;
- for each $o \in A_O$, $(B, s)Q(o)(B', s') \iff (s, s') \in R_o$ and $B' = \{t' \mid$ there exists $t \in B$ such that $(t, t') \in R_o\}$;
- for each $e \in A_E$, $(B, s)Q(e)(B', s') \iff (B, B') \in R_e$ and $s = s'$;
- $p \in V(B, s) \iff s(p) = 1$.

The intuitive idea is that each explicit state in $\mathcal{M}^C$ is a *pointed epistemic (S5) model*. The definition of $Q(o)$ is essentially the definition of progression over knowledge states for ontic action o (cf. [23]). The definition also reveals that although ontic actions are supposed to be fact-changing actions, they also affect the knowledge of the agent as B is changed to B' in a particular manner. It is clear that $\sim$ is an equivalence relation.

In this way, as one can verify, a planning problem (I, A_O, A_E, φ) in [23] for knowledge-base programs can be faithfully reduced to the model checking problem in our framework, i.e., to check whether $\mathcal{M}^C, (I, s) \vDash_{\mathbb{K}} \mathcal{K}h\varphi$ where $s \in I$. Note that the particular s does not matter since $\mathcal{K}h\varphi$ is equivalent to $\mathcal{K}\mathcal{K}h\varphi$ in our framework. It is clear that in this generalized explicit model, we have a double exponential blow-up in the number of state variables as the size of the compact model. We will come back to this in Section 6.6 about the model checking complexity.

As we mentioned, using the state-based model can help us see things that are hidden in the related definitions of the compact models. In the following, we show that the generated epistemic transition system $\mathcal{M}^C$ defined as above has the property of *perfect recall* that we mentioned in Section 6.4:

- If $(B, s)Q(e)(B', s')$ and $(B', s') \sim (B'', s'')$, it follows that $B' = B''$, $s = s'$ and $(B, B') \in R_e$ (thus $B' \subseteq B$). Thus, we have that $(B, B'') \in R_e$, and $s'' \in B' \subseteq B$. Then we have that $(B, s'')Q(e)(B'', s'')$ and $(B, s'') \sim (B, s)$.
- If $(B, s)Q(o)(B', s')$ and $(B', s') \sim (B'', s'')$, it follows that $B' = B''$ and $B' = \{t' \mid$ there exists $t \in B$ such that $(t, t') \in R_o\}$. Since (B'', s'') is in W and $B' = B''$, it follows that $s'' \in B'$. Thus, there exists $t \in B$ such that $(t, s'') \in R_o$. Hence, we have that $(B, t)Q(o)(B'', s'')$ and $(B, s) \sim (B, t)$.

On the other hand, interested readers can verify that the property of *no miracles* is not always satisfied in such an $\mathcal{M}^C$.

It is also straightforward to technically connect our models with the "half-way" models where there are explicit states with epistemic relations but only *implicit* actions, such as the ones used in planning based on Dynamic Epistemic Logic (DEL). Interested readers are referred to our previous work [31], which is based on the general approach of [54,53] treating epistemic dynamics as transitions with special properties.

6.6. On epistemic planning as model checking and its complexity

One important advantage of our framework is that we "internalize" the epistemic planning problem as a model-checking problem by using the $\mathcal{K}h$ operator in the language. For example, the plan existence problem for knowing whether p can be encoded as model checking the formula $\mathcal{K}h(\mathcal{K}p \vee \mathcal{K}\neg p)$. Moreover, our language can also express highly non-trivial nested claims about (the impossibility of) planning, e.g., $\mathcal{K}hp \wedge \neg\mathcal{K}h(p \wedge \mathcal{K}h\neg p)$ intuitively says that I have a plan to make sure p but any such plan would leave me in a position such that I do not know how to turn p off. As we mentioned before, it becomes even more interesting in a multi-agent setting. As an interesting example, $\mathcal{K}h_i(\neg\mathcal{K}h_j p \wedge \mathcal{K}_j\mathcal{K}h_i p)$ says i knows how to not let j know how to make sure p, but at the same time convince j that i knows how. As far as we know, such high-level planning problems are not discussed in the literature of planning and epistemic logic. This demonstrates the power of our know-how logic and we leave it for a future occasion for a detailed investigation.

Although model checking is not the main focus of the paper, in the rest of this subsection, for readers who are interested in complexity, we will show that model checking the full language (not just $\mathcal{K}h\varphi$) over finite models with perfect recall based on knowledge-based plans can be done in PTIME. As a quick warning, for those who are used to double-exponential complexity of contingent planning problems (such as the results in [23]), note that we are using the explicit transition systems which may require a double exponential blow-up when generated from compact representations as we showed in Section 6.5. We sketch the proof for the PTIME complexity below.

Let $\mathcal{M} = \langle W, \mathbf{A}, \sim, \{Q(a) \mid a \in \mathbf{A}\}, V\rangle$ be an epistemic transition system with perfect recall. We first focus on the core part of the model checking problem: checking whether $\mathcal{M}, w \vDash_{\mathrm{K}} \mathcal{K}h\varphi$, given that we already know where φ holds in the model. First we define a labeled transition system $T^{\mathcal{M}} = \langle D, \mathbf{A}, \{\xrightarrow{a} \mid a \in \mathbf{A}\}\rangle$ as follows:

- $D = \{[s] \mid s \in W\}$ where $[s]$ is the equivalence class of s for $\sim$;
- for each $a \in \mathbf{A}$, $[s] \xrightarrow{a} [t] \iff a$ is executable on *each* $s' \in [s]$ and there are $s' \in [s], t' \in [t]$ such that $(s', t') \in Q(a)$.

Note that this transformation can be done in PTIME in the size of $\mathcal{M}$ and the size of $T^{\mathcal{M}}$ is at most $|W|$. Intuitively, $T^{\mathcal{M}}$ is an abstraction of the original model for the purpose of deciding whether $\mathcal{K}h\varphi$. We can reduce the problem whether $\mathcal{M}, w \vDash_{\mathrm{K}} \mathcal{K}h\varphi$ to the *strong planning problem* in [12], that is, whether the planning problem $(T^{\mathcal{M}}, [w], G)$, where $[w]$ is the initial state and goal set $G = \{[s] \mid \mathcal{M}, s' \vDash_{\mathrm{K}} \varphi \text{ for all } s' \in [s]\}$, has a so-called *strong plan*, i.e., a partial function assigning each state a set of available actions on that state such that all its executions will reach some goal state in G.[15] We sketch the proof idea for the following claim:

$$\mathcal{M}, w \vDash_{\mathrm{K}} \mathcal{K}h\varphi \text{ iff there is a strong plan for } (T^{\mathcal{M}}, [w], G) \qquad (\star)$$

Suppose $\mathcal{M}, w \vDash_{\mathrm{K}} \mathcal{K}h\varphi$, then we have $\mathcal{M}, w \vDash_{\mathrm{K}} \mathcal{K}h\mathcal{K}\varphi$ since $\mathcal{M}$ satisfies perfect recall (cf. Section 6.4.) By the semantics, there is a knowledge-based plan π such that each derivation of π on each $w' \in [w]$ terminates at some state t such that $[t]$ satisfy φ in $\mathcal{M}$. With some efforts, one can show that given that π is a strongly executable knowledge-based plan and that the model satisfies perfect recall, a partial derivation $\langle \pi, w'\rangle \overset{a_1}{\Rightarrow} \langle \pi_1, w_1\rangle \cdots \overset{a_i}{\Rightarrow} \langle \pi_i, w_i\rangle$ induces a path $[w'] \overset{a_1}{\Rightarrow} [w_1] \cdots \overset{a_i}{\Rightarrow} [w_i]$ in $T^{\mathcal{M}}$. Therefore we can build an execution tree over knowledge states in D rooted at $[w]$ by using the finite derivations of π in $\mathcal{M}$. Note that the lengths of such derivations are bounded by the size of $\mathcal{M}$ and the length of π. Now we need to turn this finite execution tree into a strong plan which is a partial function assigning each $[s]$ a set of available actions. The apparent problem is that the same $[s]$ may appear multiple times in the tree but the set of actions taken at these occurrences may be different. Actually, we can unify the actions taken at each $[s]$ by using the action set of its last occurrence and this can be done inductively by some proper trimming-and-replacing of the tree from the leaves to the root. Finally we can read off the strong plan from the tree and prove that it indeed can make sure G in $T^{\mathcal{M}}$.

For the other direction, let f be a strong plan for $(T^{\mathcal{M}}, [w], G)$, we will synthesize a knowledge-based plan for $\mathcal{M}, w \vDash_{\mathrm{K}} \mathcal{K}h\varphi$. The first observation is that if f assigns $[s]$ multiple actions, then by the definition of the strong plan, we only need to select one of the actions and the plan still works. Therefore we can assume without the loss of generality that the strong plan is deterministic. Now the basic idea is simple: try to capture f by a knowledge-based plan. Note that we cannot talk about $[s]$ directly in our specification language. We can only use $\mathcal{K}\varphi$ formulas as the conditions to tell us when to execute an action. By Proposition 8, for each $[s]$, let $\mathcal{K}\varphi_{[s]}$ be the $\mathcal{L}^{\mathsf{EAL}}$-formula that characterizes $[s]$ from other $[t]$ such that $\mathcal{M}, [s] \not\leftrightarrows \mathcal{M}, [t]$. The only problematic case is when $f([s]) \neq f([t])$ for some $[s]$ and $[t]$ such that $\mathcal{M}, [s] \leftrightarrows \mathcal{M}, [t]$ thus you cannot use $\mathcal{K}\varphi$ formulas to distinguish them. However, by the idea behind the proof of Proposition 27, we can show that if $\mathcal{M}, [s_0] \leftrightarrows \mathcal{M}, [t_0]$ and $[s_0] \xrightarrow{a_1} [s_1] \cdots \xrightarrow{a_k} [s_k]$ in $T^{\mathcal{M}}$, then there are $t_1, \cdots, t_k$ such that $[t_0] \xrightarrow{a_1} [t_1] \cdots \xrightarrow{a_k} [t_k]$ in $T^{\mathcal{M}}$ and $\mathcal{M}, [s_i] \leftrightarrows \mathcal{M}, [t_i]$ for all $0 \leq i \leq k$. Note that given $\mathcal{M}, [s_k] \leftrightarrows \mathcal{M}, [t_k]$, there is a $t' \in t_k$ such that $\mathcal{M}, s_k \leftrightarrows \mathcal{M}, t'$. Now by Theorem 28 $\mathcal{M}, s_k \vDash \mathcal{K}\varphi$ iff $\mathcal{M}, t' \vDash \mathcal{K}\varphi$ iff $\mathcal{M}, t_k \vDash \mathcal{K}\varphi$ thus $[s_k] \in G$ iff $[t_k] \in G$. Therefore for any bisimilar pair $[s]$ and $[t]$ we can always choose one of the actions say $f([s])$ and make sure the resulting execution reaches G. In the other words,

[15] In [12] a plan is represented as a collection of pairs of $\langle state, action\rangle$.

Algorithm 1: A model checking procedure.

```
Procedure MC(M, w, φ):                          /* check whether M, w ⊨_K φ */
    switch φ do                                 /* Boolean cases are omitted */
        case Kφ do
            foreach w' ∈ [w] do
                if not MC(M, w', φ) then return false ;
            return true;
        case Khφ do
            G := ∅;
            foreach [s] in T^M do
                if MC(M, s, Kφ) then G := G ∪ {[s]} ;
            return StongPlanExistence(T^M, [w], G);
```

given f we can find a strong plan f' such that $f'([s]) = f'([t])$ for bisimilar $[s]$ and $[t]$. Now, to define the knowledge-based program π based on f', we first define π_i for each $1 \leq i \leq n$:

$$\pi_1 := \texttt{if } \mathcal{K}\varphi_{[s_1]} \texttt{ then } f'([s_1]) \texttt{ else } \epsilon;$$

$$\text{for each } j < i,\ \pi_{j+1} := \texttt{if } \mathcal{K}\varphi_{[s_{j+1}]} \texttt{ then } f'([s_{j+1}]) \texttt{ else } \pi_j$$

The knowledge-based program π is defined as $\pi = \texttt{while } (\mathcal{K}\varphi_{[s_1]} \vee \cdots \vee \mathcal{K}\varphi_{[s_n]}) \texttt{ do } \pi_n$. Then with some efforts we can verify that π is a good plan to justify $\mathcal{M}, w \models_{\mathrm{K}} \mathcal{K}h\varphi$, which completes the sketch of the proof for the Claim ($\star$).

Now we present the model checking algorithm MC in Algorithm 1 for $\models_{\mathrm{K}}$ on epistemic transition systems with perfect recall by calling StrongPlanExistence, the procedure for solving the corresponding strong planning problem $(T^{\mathcal{M}}, [w], G)$. Based on the algorithm in [12], using some fixed point computation from G backwards, StrongPlanExistence can be done in PTIME of the size of $T^{\mathcal{M}}$, and the size of $T^{\mathcal{M}}$ is bounded by the size of $\mathcal{M}$, it follows that Algorithm 1 is in PTIME in the size of $\mathcal{M}$.

The above analysis also shows that considering the full language instead of the $\mathcal{K}h\varphi$ formulas does not really increase the computational complexity. So to some extent, we can handle the more general planning problems for free, as we also observed in the case of the more restrictive conformant planning in [32] using a model checking method.

We leave the full complexity analysis for model checking with respect to other semantics of $\mathcal{K}h$ with or without the prefect recall properties in a multi-agent setting for the future work.

6.7. Model theory and proof theory of epistemic planning

Since we have internalized the epistemic planning as formulas in a logical language, natural model theoretical and proof theoretical questions for planning also arise.

Model theory A natural question to ask in epistemic planning is when two different models can be viewed the same for planning? The counterpart question in model theory of modal logic is usually answered by using some notion of bisimulation. Indeed, we have already seen the notion of bisimulation (Definition 6) played important roles in various proofs for completeness and for model checking. Theorem 28 tells us that ELKh is invariant under bisimulation but it is not yet the exact match of the logical equivalence of ELKh, and we can make it coarser as in the case of other know-how logics as discussed in [17]. With such notions, we can try to make the models smaller even before doing planning. We can also ask how to compare the "planning power" of different types of plans, in terms of the logical strength of $\mathcal{K}h\varphi$ formulas. Proposition 22 is such an example of using model theoretical idea to study the relative expressive power of various notions of planning. We believe there are many questions of this kind to be answered.

Proof theory Our axiomatization result is not merely philosophical. It not only shows that the core axioms are shared by all those ten types of planning notions in terms of know-how, but also provides a syntactic method to do high-level reasoning about planning in terms of know-how, which can be automated in principle based on the decidability. This is actually the more traditional way of using logic than model checking that we discussed in the previous subsection. The axioms and rules tell us which reasoning patterns are good or bad (cf. Proposition 29). As we mentioned in Section 6.4, further assumptions about the model will bring new axioms and give us more power in reasoning. An interesting type of such reasoning may have the form: this know-how leads to that know-how. We may develop this approach further by using more "practical" Gentzen-style systems to make the syntactic reasoning more easily to be implemented. We leave it also to a future occasion.

7. Conclusions

In this paper, we present a unified logical framework for planning-based knowing how. Different notions of plans are captured by sublanguages of a plan specification language with a rigorous semantics. We show that different kinds of know-

how semantics based on various types of plans share exactly the same logic. An extension with an intuitive composition axiom axiomatizes the logic over finite models based on knowledge-based plans. We also show that the two logics are decidable.

In the context of planning, our first unified axiomatization result also suggests that the generic notion of epistemic planning, as we formalized for the semantics of $\mathcal{K}h$, is quite robust, which is independent of the specific notion of plans, at least in the most general case without further assumption about the models. By using the $\mathcal{K}h$ modality we can also internalize epistemic planning in the logical language. This gives us the advantage to generalize the usual planning problems greatly by using $\mathcal{K}h$, $\mathcal{K}$ and their nesting and Boolean combinations. We can then use model checking techniques to handle such generalized planning problems. As we see in Section 6.6, model checking the generalized problems does not increase the complexity in our case study. We can also use the proof system to do syntactic high-level reasoning of planning without fixing the model. Finally, this logical approach to epistemic planning can also bring the theoretical tools such as bisimulation on the radar of AI planners. We may use the knowledge of logic to help us to plan better.

As we already mentioned in Section 6, there are many immediate directions to go:

- A dynamified version of our know-how logic over compact representations of actions directly. This may bring us closer to the standard practice of planning in AI.
- The logics of know-how given various extra properties such as perfect recall and no miracles discussed in Section 6.4.
- The model checking problem of our logic based on various plans and various conditions of the model in a *multi-agent setting*.
- It is also interesting to consider know-how logics based on probabilistic programs and probabilistic planning as in [24,8,29,40].
- Finally, it would be interesting to develop a group-based know-how framework to accommodate non-trivial multi-step plans specified by our language `Prg`.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

The authors thank Valentin Goranko for in-depth discussions on using PDL to capture plans, which partly inspired this work. Yanjun Li acknowledges the support from the National Social Science Foundation for Young Scholars of China (Grant No. 18CZX062). The authors are grateful to the anonymous reviewers of this journal, whose critical comments helped to improve the paper.

References

[1] Thomas Ågotnes, Natasha Alechina, Coalition logic with individual, distributed and common knowledge, J. Log. Comput. 29 (2019) 1041–1069.
[2] Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, Alternating-time temporal logic, J. ACM 49 (2002) 672–713.
[3] Mikkel Birkegaard Andersen, Thomas Bolander, Martin Holm Jensen, Conditional epistemic planning, in: Logics in Artificial Intelligence, Springer, 2012, pp. 94–106.
[4] Mikkel Birkegaard Andersen, Thomas Bolander, Martin Holm Jensen, Don't plan for the unexpected: planning based on plausibility models, Log. Anal. 58 (2015).
[5] Guillaume Aucher, DEL-sequents for regression and epistemic planning, J. Appl. Non-Class. Log. 22 (4) (2012) 337–367.
[6] Guillaume Aucher, Thomas Bolander, Undecidability in epistemic planning, in: Proceedings of IJCAI '13, 2013, pp. 27–33.
[7] Jos Baeten, A brief history of process algebra, Theor. Comput. Sci. 335 (2–3) (2005) 131–146.
[8] Vaishak Belle, Probabilistic planning by probabilistic programming, in: Proceedings of Workshops of AAAI '18, 2018, pp. 654–657.
[9] Patrick Blackburn, Maarten de Rijke, Yde Venema, Modal Logic, Cambridge University Press, 2002.
[10] Thomas Bolander, Mikkel Birkegaard Andersen, Epistemic planning for single and multi-agent systems, J. Appl. Non-Class. Log. 21 (1) (2011) 9–34.
[11] Thomas Bolander, Martin Holm Jensen, François Schwarzentruber, Complexity results in epistemic planning, in: Proceedings of IJCAI '15, 2015, pp. 2791–2797.
[12] A. Cimatti, M. Pistore, M. Roveri, P. Traverso, Weak, strong, and strong cyclic planning via symbolic model checking, Artif. Intell. 147 (1–2) (July 2003) 35–84.
[13] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, Moshe Y. Vardi, Knowledge-based programs, Distrib. Comput. 10 (4) (1997) 199–225.
[14] Jie Fan, Yanjing Wang, Hans van Ditmarsch, Contingency and knowing whether, Rev. Symb. Log. 8 (01) (2015) 75–107.
[15] Jeremy Fantl, Knowing-how and knowing-that, Philos. Compass 3 (3) (2008) 451–470.
[16] Raul Fervari, Andreas Herzig, Yanjun Li, Yanjing Wang, Strategically knowing how, in: Proceedings of IJCAI '17, 2017, pp. 1031–1038.
[17] Raul Fervari, Fernando R. Velázquez-Quesada, Yanjing Wang, Bisimulations for knowing how logics, in: Rev. Symb. Log., 2021, in press.
[18] Hector Geffner, Blai Bonet, A Concise Introduction to Models and Methods for Automated Planning, Morgan & Claypool Publishers, 2013.
[19] Malik Ghallab, Dana Nau, Paolo Traverso, Automated Planning: Theory and Practice, Elsevier, 2004.
[20] Andreas Herzig, Logics of knowledge and action: critical analysis and challenges, Auton. Agents Multi-Agent Syst. 29 (5) (2015) 719–753.
[21] Martin Holm Jensen, Epistemic and Doxastic Planning, PhD thesis, Technical University of Denmark, 2014.
[22] Jérôme Lang, Bruno Zanuttini, Knowledge-based programs as plans - the complexity of plan verification, in: ECAI '12, 2012, pp. 504–509.
[23] Jérôme Lang, Bruno Zanuttini, Knowledge-based programs as plans: succinctness and the complexity of plan existence (extended abstract), in: TARK '13, 2013.

[24] Jérôme Lang, Bruno Zanuttini, Probabilistic knowledge-based programs, in: Proceedings of IJCAI '15, 2015, pp. 1594–1600.
[25] Tszyuen Lau, Yanjing Wang, Knowing your ability, Philos. Forum 3–4 (2016) 415–424.
[26] Hector J. Levesque, What is planning in the presence of sensing?, in: Proceedings of the 13th National Conference on Artificial Intelligence, AAI, 1996.
[27] Hector J. Levesque, Planning with loops, in: Proceedings of IJCAI '05, 2005, pp. 509–515.
[28] Yanjun Li, Stopping means achieving: a weaker logic of knowing how, Stud. Log. 9 (4) (2016) 34–54.
[29] Yanjun Li, Barteld Kooi, Yanjing Wang, A dynamic epistemic framework for reasoning about conformant probabilistic plans, Artif. Intell. 268 (2019) 54–84.
[30] Yanjun Li, Yanjing Wang, Achieving while maintaining: a logic of knowing how with intermediate constraints, in: Proceedings of ICLA'17, 2017, pp. 154–167.
[31] Yanjun Li, Yanjing Wang, Multi-agent knowing how via multi-step plans: a dynamic epistemic planning based approach, in: Proceedings of LORI VII, 2019, pp. 126–139.
[32] Yanjun Li, Quan Yu, Yanjing Wang, More for free: a dynamic epistemic framework for conformant planning over transition systems, J. Log. Comput. 27 (8) (2017) 2383–2410.
[33] Benedikt Löwe, Eric Pacuit, Andreas Witzel, DEL planning and some tractable cases, in: Proceedings of LORI III, Springer, 2011, pp. 179–192.
[34] John McCarthy, First order theories of individual concepts and propositions, Mach. Intell. 9 (1979) 129–147.
[35] Robert C. Moore, A formal theory of knowledge and action, Technical report, DTIC Document, 1984.
[36] Christian Muise, Vaishak Belle, Paolo Felli, Sheila McIlraith, Tim Miller, Adrian R. Pearce, Liz Sonenberg, Planning over multi-agent epistemic states: a classical planning approach, in: Proceedings of AAAI '15, 2015.
[37] Pavel Naumov, Jia Tao, Together we know how to achieve: an epistemic logic of know-how (extended abstract), in: Proceedings of TARK '17, vol. 251, 2017, pp. 441–453.
[38] Pavel Naumov, Jia Tao, Second-order know-how strategies, in: Proceedings of AAMAS '18, 2018, pp. 390–398.
[39] Pavel Naumov, Jia Tao, Together we know how to achieve: an epistemic logic of know-how, Artif. Intell. 262 (2018) 279–300.
[40] Pavel Naumov, Jia Tao, Knowing-how under uncertainty, Artif. Intell. 276 (2019) 41–56.
[41] Hanne Riis Nielson, Flemming Nielson, Semantics with Applications: An Appetizer, Springer, 2007.
[42] Pere Pardo, Mehrnoosh Sadrzadeh, Strong planning in the logics of communication and change, in: Declarative Agent Languages and Technologies X, Springer, 2013, pp. 37–56.
[43] Pauly Marc, A modal logic for coalitional power in games, J. Log. Comput. 12 (1) (2002) 149–166.
[44] David E. Smith, Daniel S. Weld, Conformant graphplan, in: Proceedings of AAAI '98, 1998, pp. 889–896.
[45] Jason Stanley, Know How, Oxford University Press, 2011.
[46] Jason Stanley, Timothy Williamson, Knowing how, J. Philos. 98 (2001) 411–444.
[47] Wiebe van der Hoek, Michael Wooldridge, Cooperation, knowledge, and time: alternating-time temporal epistemic logic and its applications, Stud. Log. (2003) 125–157.
[48] Xun Wang, A logic of knowing how with skippable plans, in: Proceedings of LORI-VII, 2019, pp. 413–424.
[49] Yanjing Wang, A logic of knowing how, in: Proceedings of LORI-V, 2015, pp. 392–405.
[50] Yanjing Wang, Representing imperfect information of procedures with hyper models, in: Proceedings of ICLA '15, vol. 8923, 2015.
[51] Yanjing Wang, A logic of goal-directed knowing how, Synthese 195 (2018) 4419–4439.
[52] Yanjing Wang, Beyond knowing that: a new generation of epistemic logics, in: Jaakko Hintikka on Knowledge and Game Theoretical Semantics, Springer, Cham, 2018, pp. 499–533.
[53] Yanjing Wang, Guillaume Aucher, An alternative axiomatization of DEL and its applications, in: Proceedings of IJCAI '13, 2013, pp. 1147–1154.
[54] Yanjing Wang, Qinxiang Cao, On axiomatizations of public announcement logic, Synthese 190 (1S) (2013) 103–134.
[55] Yanjing Wang, Jie Fan, Conditionally knowing what, in: Advances in Modal Logic, vol. 10, 2014, pp. 569–587.
[56] Yanjing Wang, Yanjun Li, Not all those who wander are lost: dynamic epistemic reasoning in navigation, in: Advances in Modal Logic, vol. 9, 2012, pp. 559–580.
[57] Peipei Wu, Yanjun Li, A logic for multi-agent conformant planning over transition systems, IEEE Access 8 (2020) 193621–193631.
[58] Chao Xu, Yanjing Wang, Thomas Studer, A logic of knowing why, Synthese 198 (2021) 1259–1285, https://doi.org/10.1007/s11229-019-02104-0.
[59] Quan Yu, Yanjun Li, Yanjing Wang, A dynamic epistemic framework for conformant planning, in: Proceedings of TARK '15, vol. 215, 2015, pp. 249–259.
[60] Quan Yu, Ximing Wen, Yongmei Liu, Multi-agent epistemic explanatory diagnosis via reasoning about actions, in: Proceedings of IJCAI'13, 2013, pp. 1183–1190.