Interpreting Knowledge-based Programs

Alexander Knapp[1](✉), Heribert Mühlberger[1], and Bernhard Reus[2]

[1] Universität Augsburg, Augsburg, Germany
{knapp, muehlber}@informatik.uni-augsburg.de
[2] University of Sussex, Brighton, UK
bernhard@sussex.ac.uk

Abstract Knowledge-based programs specify multi-agent protocols with epistemic guards that abstract from how agents learn and record facts or information about other agents and the environment mutual dependency between the evaluation of epistemic guards over the reachable states and the derivation of the reachable states depending on the evaluation of epistemic guards synchronous programming languages to the interpretation problem of knowledge-based programs and demonstrate that the resulting constructive interpretation is monotone and has a least fixed point. We relate our approach with existing interpretation schemes for both synchronous and asynchronous programs interpretation and illustrate the procedure by several examples and an application to the Java memory model.

1 Introduction

Knowledge-based programs [14] describe multi-agent systems based on explicit knowledge tests on what an agent knows or does not know about itself, other agents, and the environment: Extending standard programs, an agent may look beyond what it can directly observe by reasoning about the possible states of the other agents and the environment in all possible program executions. Such non-local, epistemic conditions abstract from how an agent may learn and record particular environmental facts or information about other agents. Thus knowledge-based programs rather are specifications of (multi-agent) protocols that may be implemented by standard, directly executable programs. For being implementable in the first place, however, it has to be ensured that the knowledge guards can be resolved consistently given all possible program executions.

Consider for example a bit transmission [14, Ex. 4.1.1, Ex. 7.1.1], where a sender S has to transmit a bit sbit over a lossy channel to a receiver R who has to acknowledge the reception, again over a lossy channel. This can be modelled by a knowledge-based program over the state variables $\mathrm{sbit} \in \{0,1\}$, $\mathrm{rval} \in \{\bot,0,1\}$, and $\mathrm{ack} \in \{0,1\}$ as follows: S can only directly observe (read) sbit and ack, and R only rval (but both may write all variables); $(\mathsf{K}_\mathrm{R}\,\mathrm{sbit} = 0) \vee (\mathsf{K}_\mathrm{R}\,\mathrm{sbit} = 1)$ expresses that R knows sbit's value and is abbreviated by $\mathsf{K}_\mathrm{R}\,\mathit{sbit}$. The behaviour description consists of a looping guarded command with two branches that is started with $\mathrm{rval} = \bot$ and $\mathrm{ack} = 0$, but sbit left undetermined:

$$\begin{array}{ll} \textbf{do}\ \neg\mathsf{K}_\mathrm{S}\,\mathsf{K}_\mathrm{R}\,\mathit{sbit} \twoheadrightarrow (\mathrm{rval} \leftarrow \mathrm{sbit}\ \texttt{or skip}) & \text{— S} \\ \quad [\!]\ \mathsf{K}_\mathrm{R}\,\mathit{sbit} \wedge \neg\mathsf{K}_\mathrm{R}\,\mathsf{K}_\mathrm{S}\,\mathsf{K}_\mathrm{R}\,\mathit{sbit} \twoheadrightarrow (\mathrm{ack} \leftarrow 1\ \texttt{or skip})\ \textbf{od} & \text{— R} \end{array}$$

The guarded branches are separated by a $[\!]$, `or` means a non-deterministic choice, and `skip` doing nothing: S sends the bit as long as it does not know that R received it, and R

T. Wies (Ed.): ESOP 2023, LNCS 13990, pp. 253–280, 2023.
https://doi.org/10.1007/978-3-031-30044-8_10

keeps acknowledging once it has learnt the bit and does not know that S knows this fact. The epistemic formulæ $\mathsf{K}_a\,\varphi$ in the program are to be interpreted as in classical Kripke semantics: φ holds in all states (or worlds) that agent a currently deems possible. Which states these are is regulated on the one hand by what a can observe: any state that is indistinguishable from the current one by the available observations is possible for the agent. In the example only S can observe sbit, though, due to the protocol, it should be possible that eventually R knows its value. On the other hand, the possible states depend on which runs of the knowledge-based program may actually happen, i.e., which states are reachable taking epistemically guarded transitions: If only the actions of the program are taken, it is impossible to reach a state satisfying both $\mathrm{rval} \neq \bot$ and $\mathrm{rval} \neq \mathrm{sbit}$, which, however, is present in the global state space; but it is decisive that it is not reachable in any execution in order to have some execution where $\mathsf{K}_\mathrm{R}\ \mathit{sbit}$ can become true.

The interpretation of knowledge-based programs hinges precisely on this mutual dependency between the evaluation of epistemic guards over the reachable states and the derivation of the reachable states depending on the evaluation of the epistemic guards. This implicit definition of the epistemic state of the agents by the observables and the reachable states of the commonly known protocol is in stark contrast to Baltag's epistemic action models [4,31], where the epistemic state is given and manipulated explicitly. In many cases, including the bit transmission protocol, the reachable state space may be computed using static analysis techniques without taking into account the epistemic nature of the guards. However, the interplay between knowledge and reachability may sometimes become more intricate: The more states are reachable the less is known definitely, and the guards will in turn influence what is reachable positively or negatively.

Consider, for another example, a variable setting problem [14, Exc. 7.5] involving a single agent a and a single state variable $\mathrm{x} \in \{0,1,2,3\}$, where a cannot observe x directly. The agent executes the following guarded command starting with $\mathrm{x} = 0$:

$$\begin{array}{l}\textbf{if}\ \mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 1 \twoheadrightarrow \mathrm{x} \leftarrow 3\\ \quad[\!]\ \mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 3 \twoheadrightarrow \mathrm{x} \leftarrow 1\ \textbf{fi}\end{array}$$

Being an initial condition, $\mathrm{x} = 0$ is reachable, whereas $\mathrm{x} = 2$ is not reachable as 2 is never assigned. However, two different sets of reachable states make for a consistent interpretation of the knowledge guards for the remaining values: $\{\mathrm{x} = 0, \mathrm{x} = 1\}$, where $\mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 1$ is false and $\mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 3$ is true, and $\{\mathrm{x} = 0, \mathrm{x} = 3\}$, with the opposite results. The singleton set $\{\mathrm{x} = 0\}$ is ruled out, since both guards would be true such that $\mathrm{x} = 3$ and $\mathrm{x} = 1$ are reachable; and $\{\mathrm{x} = 0, \mathrm{x} = 1, \mathrm{x} = 3\}$ is impossible, since both guards are false and thus neither $\mathrm{x} = 1$ nor $\mathrm{x} = 3$ are reachable. Breaking this cycle by making one of the transitions unconditional on knowledge as, e.g., in

$$\begin{array}{l}\textbf{if}\ \mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 1 \twoheadrightarrow \mathrm{x} \leftarrow 3\\ \quad[\!]\ \mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 3 \twoheadrightarrow \mathrm{x} \leftarrow 2\\ \quad[\!]\ \mathrm{true} \twoheadrightarrow \mathrm{x} \leftarrow 1 \qquad \textbf{fi}\end{array}$$

yields a knowledge-based program with the unique consistent interpretation $\{\mathrm{x} = 1, \mathrm{x} = 2\}$. For computing its behaviour, however, several steps are needed, first reasoning that $\mathrm{x} = 1$ is reachable, then that $\mathrm{x} = 3$ is not reachable, and, finally, that $\mathrm{x} = 2$ is reachable.

Related Work. In their introduction and seminal treatise on knowledge-based programs [13,14], Fagin et al. characterise the unique interpretability of such programs by their "dependence on the past" w. r. t. some non-empty class of transition systems: The evaluation of knowledge guards in a state coincides for all interpretations in the class that share a common past of the state. A sufficient condition for this dependence is that the program "provides epistemic witnesses" for all interpretations of the class such that not knowing something at some point in time has a counter example in the past. A sufficient condition for this provision, in turn, is that the program is "synchronous", i. e., that all agents can determine the global time from their local states. For example, the bit transmission protocol provides epistemic witnesses and thus is uniquely interpretable; but it is not synchronous. The cycle-breaking variable setting program is also uniquely interpretable, but does not provide epistemic witnesses. For "asynchronous" knowledge-based programs, De Haan et al. [10] suggest to rely on classical iteration of the non-monotone reachability functional that interprets the knowledge modalities according to what currently is assumed to be reachable. The computation process is started with all states assumed to be reachable and stops when some set of states is repeated. This approach fixes some semantics for all knowledge-based programs, also for those which are cyclic and contradictory or only self-fulfilling.

The problem of mutual dependence of guard evaluation and reachability has also occurred in the design of synchronous programming languages [6] for embedded systems, like Esterel [7] or Lustre [18], which rely on "perfect synchrony": a step for reacting to some inputs takes zero time and output signals are produced at exactly the same time as the input signals. Since thus the status of a signal to be produced can be queried at the same time, this requires "logical coherence" saying that a (non-input) signal is present in a step of execution if, and only if, a command emitting this signal is executed in this step. Whereas Lustre forbids cyclic programs on a syntactic basis, Berry's approach to the semantics of Esterel [8] singles out "reactive" — at least one execution — and "determinate" — at most one execution — programs using a static executability analysis: It is computed which signals *must* be present, i. e., have to occur inevitably, and which signals *cannot* be present, i. e., have no emitting execution. This is also referred to as must/cannot analysis and has to be performed several times for finding a fixed point of all the signal statuses.

In logic programming involving "negation as failure" under- and over-approximations in terms of three- and four-valued logics lead to the "Kripke-Kleene fixpoint" and "well-founded" models; see [11] for an overview. There, however, the temporal dimension of reachability or executability is not involved. The "stable model semantics" [16,5] stresses the rational inclusion or exclusion of atoms: A set of atoms M is "stable" for a logic program Π if it coincides with the minimal set of atoms inferable from the "reduct" Π_M which is obtained from Π by deleting each clause that has a negative literal $\neg p$ in its body with $p \in M$, and all negative literals in the bodies of the remaining clauses. The definition is not algorithmic or constructive; the minimality condition rules out self-fulfilling solutions, the reduction process avoids contradictions. Gelfond's "epistemic specifications" [15] extend (disjunctive) logic programs with a modality K for "subjective literals" for representing incomplete information in programs with several stable models.

Contributions. We apply the principles of the must/cannot analysis to the interpretability problem of knowledge-based programs. After recalling some basic notions of epistemic logic and epistemic transition structures (Sect. 2), we first recapitulate the approaches

by Fagin et al. [14] and De Haan et al. [10] in terms of epistemically guarded transition systems, a syntax-agnostic format for knowledge-based programs (Sect. 3). For a more direct analysis, our account of those designs is state-based rather than run-based. We demonstrate the results and the limits of both interpretation schemes by several examples that illustrate (a-)synchronicity and non-monotone interpretation for cyclic, contradictory, or self-fulfilling programs. The latter behaviour is the main motivation for our reformulation of the interpretation problem in terms of epistemic must/can transition structures which offer lower and upper bounds on the behaviour of a knowledge-based program (Sect. 4). We show that this constructive interpretation is always monotone and yields a least fixed point. However, lower and upper bound of the fixed point need not always coincide and we relate decided fixed points with the notions of "providing epistemic witnesses" and synchronicity. We then derive a representation of the behaviour of a knowledge-based program as a general rule system with not only positive but also negative premisses (Sect. 5). Such rule systems correspond to logic programs involving "negation as failure" and the intended solutions form "stable models". The must/can approximation technique, its monotonicity, and it fixed point properties directly transfer to such rule systems. We finally describe an implementation of our constructive interpretation approach in the "Temporal Epistemic Model Interpreter and Checker" (TEMIC, Sect. 6). For model checking interpreted knowledge-based programs, the tool supports CTLK, the combination of "Computational Tree Logic" (CTL) with epistemic logic. Moreover, this logic can also be used in program guards; the interpretation of such temporal-epistemic programs extends the previous approaches. We give some applications to the analysis of the Java memory model.

2 Epistemic Logic and Epistemic Transition Structures

We briefly summarise the basic notions of epistemic logic for expressing knowledge guards [31,30]. We then define epistemic transition structures as the domain of interpretation of knowledge-based programs. These transition structures combine the temporal dimension of executing a program with the epistemic dimension for evaluating what agents know. Both the logic and the transition structures are built over an *epistemic signature* $\Sigma = (P, A)$ that consists of a set of *propositions* P and a set of *agents* A.

2.1 Epistemic Logic

An *epistemic structure* $K = (W, R, L)$ over (P, A) is given by a set of *worlds* W, an A-family of epistemic *accessibility relations* $R = (R_a \subseteq W \times W)_{a \in A}$, and a *labelling* $L\colon W \to \wp P$ assigning each world a set of propositions. In concrete examples, we will require R_a to be an equivalence relation such that if $(w_1, w_2) \in R_a$, then agent a cannot distinguish between the two worlds w_1 and w_2. The *epistemic formulæ* $\varphi \in \Phi_{P,A}$ over (P, A) are defined by the following grammar:

$$\varphi ::= p \mid \text{false} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathsf{K}_a\, \varphi$$

where $p \in P$ and $a \in A$. The epistemic formula $\mathsf{K}_a\,\varphi$ is to be read as "agent a *knows* φ". We use the usual propositional abbreviations true for $\neg\mathrm{false}$ and $\varphi_1 \vee \varphi_2$ for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$. Furthermore, we consider the epistemic modality M as the dual of K, such that $\mathsf{M}_a\,\varphi$ abbreviates $\neg\mathsf{K}_a\,\neg\varphi$ and is to be read as "agent a *deems* φ *possible*". The *satisfaction relation* of an epistemic formula $\varphi \in \Phi_{P,A}$ over an epistemic structure $K = (W, R, L)$ over (P, A) at a world $w \in W$, written $K, w \models \varphi$, is inductively defined by

$$
\begin{aligned}
&K, w \models p \iff p \in L(w)\\
&K, w \not\models \mathrm{false}\\
&K, w \models \neg\varphi \iff K, w \not\models \varphi\\
&K, w \models \varphi_1 \wedge \varphi_2 \iff K, w \models \varphi_1 \text{ and } K, w \models \varphi_2\\
&K, w \models \mathsf{K}_a\,\varphi \iff K, w' \models \varphi \text{ f. a. } w' \in W \text{ with } (w, w') \in R_a
\end{aligned}
$$

2.2 Epistemic Transition Structures

An epistemic transition structure combines a temporal transition relation with an epistemic accessibility relation over a common set of states. The transitions describe which states can be reached from a set of initial states, the accessibilities specify which states are indistinguishable. Knowledge formulæ are evaluated over the associated global epistemic structure. This derived structure has the reachable states as its worlds and reuses the accessibility relation and the labelling but restricted to the reachable states.

Formally, an *epistemic transition structure* $M = (S, E, L, S_0, T)$ over (P, A) is given by an epistemic structure (S, E, L), a set of temporally *initial states* $S_0 \subseteq S$, and a temporal *transition relation* $T \subseteq S \times S$. We write $S(M)$ for S, $T(M)$ for T, etc. The (temporally) *reachable states* $S_\omega(M) = \bigcup_{0 \le k} S_k(M)$ and *transition relation* $T_\omega(M) = \bigcup_{0 \le k} T_k(M)$ of M are inductively defined by

$$
\begin{aligned}
&S_0(M) = S_0, \quad S_{k+1}(M) = S_k(M) \cup \{s' \mid \text{ex. } s \in S_k(M) \text{ s. t. } (s, s') \in T\}\,;\\
&T_0(M) = \emptyset, \quad T_{k+1}(M) = T_k(M) \cup \{(s, s') \in T \mid s \in S_k(M)\}\,.
\end{aligned}
$$

The associated *epistemic structure* of M is given by

$$K(M) = (S_\omega(M), E \cap S_\omega(M)^2, L{\restriction}S_\omega(M))$$

where $S_\omega(M)^2$ abbreviates $S_\omega(M) \times S_\omega(M)$ and $L{\restriction}S_\omega(M)$ denotes labelling L restricted to domain $S_\omega(M)$. The *satisfaction relation* of an epistemic formula $\varphi \in \Phi_{P,A}$ over M at an $s \in S_\omega(M)$, written $M, s \models \varphi$, is defined as

$$M, s \models \varphi \iff K(M), s \models \varphi\,.$$

The set of epistemic transition structures over $\Sigma = (P, A)$ sharing the same *epistemic state basis* $\mathsf{B} = (S, E, L, S_0)$ is denoted by $\mathscr{M}_\Sigma(\mathsf{B})$. We say that $M_1 \subseteq M_2$ for $M_1, M_2 \in \mathscr{M}_\Sigma(\mathsf{B})$ if $T(M_1) \subseteq T(M_2)$ and similarly extend union and intersection from transition relations to epistemic transition structures.

3 Knowledge-based Programs

Knowledge-based programs extend standard programs by explicit knowledge tests. Their interpretation involves a cycle: the evaluation of the epistemic guards depends on the program's reachable states, the derivation of the reachable states on the evaluation of the program's epistemic guards.

We render knowledge-based programs in a syntax-agnostic format as epistemically guarded transition systems. Like epistemic transition structures, these guarded systems operate on a global set of states with epistemic accessibilities and a propositional labelling. All program steps are represented as knowledge-guarded actions of the form $\varphi \supset B$ with φ an epistemic formula and B a relation on the semantic states. Knowledge-independent decisions are obtained by choosing $\varphi = \mathrm{true}$, and any kind of program control structure can be expressed by a judicious choice of guarded actions.

Breaking up the cyclic step of assigning meaning to a knowledge-based program, an epistemically guarded transition system Γ is interpreted over an epistemic transition structure M yielding another epistemic transition structure Γ^M. A guarded action $\varphi \supset B$ of Γ contributes those $(s, s') \in B$ for which $M, s \models \varphi$, where, in particular, s is reachable in M. What is sought for is a consistent interpretation with $\Gamma^M = M$ such that reachability and knowledge are mutually justified. Finding such a balanced structure is complicated by the fact that the interpretation functional is not monotone in general: The more is reachable the less is known and this may make more or less states reachable.

After introducing and illustrating our format of knowledge-based programs we summarise and adapt two existing approaches to their interpretation that have been devised for run-based rather than state-based systems: De Haan et al. [10] propose to iterate the interpretation functional starting from an epistemic transition structure where all states are reachable. Iteration stops when either a fixed point is reached or, due to non-monotonicity, a contradiction is found. In this way all knowledge-based programs are assigned some semantics and there is no distinction between meaningful and contradictory or just self-fulfilling programs. The original approach by Fagin et al. [13,14] characterises knowledge-based programs that admit a unique consistent interpretation by the notion of dependence on the past. A sufficient condition of providing epistemic witnesses is developed which, in particular, applies to the subclass of synchronous knowledge-based programs.

3.1 Epistemically Guarded Transition Systems

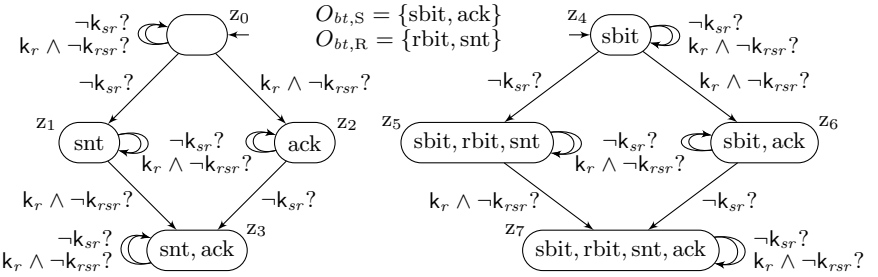
An *epistemically guarded transition system* $\Gamma = (S, E, L, S_0, \mathcal{T})$ over (P, A) is given by an epistemic state basis (S, E, L, S_0) over (P, A) and a set $\mathcal{T}$ of *epistemically guarded actions* $\varphi \supset B$ consisting of an epistemic formula $\varphi \in \Phi_{P,A}$ as *guard* and a transition relation $B \subseteq S \times S$.

Example 1. (a) Consider the bit transmission problem of the introduction:

$$\begin{array}{l}\textbf{do}\ \neg \mathsf{K}_\mathrm{S}\, \mathsf{K}_\mathrm{R}\, \mathit{sbit} \twoheadrightarrow (\mathrm{rval} \leftarrow \mathrm{sbit}\ \texttt{or}\ \texttt{skip}) \\ \quad [\!]\ \mathsf{K}_\mathrm{R}\, \mathit{sbit} \wedge \neg \mathsf{K}_\mathrm{R}\, \mathsf{K}_\mathrm{S}\, \mathsf{K}_\mathrm{R}\, \mathit{sbit} \twoheadrightarrow (\mathrm{ack} \leftarrow 1\ \texttt{or}\ \texttt{skip})\ \textbf{od}\end{array}$$

A sender agent S sends a bit $\mathrm{sbit} \in \{0, 1\}$ to a receiver agent R over an unreliable channel by setting $\mathrm{rval} \in \{\bot, 0, 1\}$; and R acknowledges the reception over an unreliable channel by setting $\mathrm{ack} \in \{0, 1\}$. Again, we abbreviate $(\mathsf{K}_\mathrm{R}\, \neg\mathrm{sbit}) \vee (\mathsf{K}_\mathrm{R}\, \mathrm{sbit})$ expressing that

the receiver knows the bit to be sent by $\mathsf{K}_\mathrm{R}\ \mathit{sbit}$. We concretise the problem into an epistemically guarded transition system $\Gamma_{bt} = (\mathsf{B}_{bt}, \mathcal{T}_{bt})$ with $\mathsf{B}_{bt} = (S_{bt}, E_{bt}, L_{bt}, S_{bt,0})$ over $\Sigma_{bt} = (P_{bt}, A_{bt})$ with $P_{bt} = \{\mathrm{sbit}, \mathrm{rbit}, \mathrm{snt}, \mathrm{ack}\}$ and $A_{bt} = \{\mathrm{S}, \mathrm{R}\}$. Since we use a propositional encoding, we represent $\mathrm{rval} \in \{\bot, 0, 1\}$ by a proposition rbit for the transmitted bit and a proposition snt for the validity of rbit. Further abbreviating the knowledge guards $\mathsf{K}_\mathrm{R}\ \mathit{sbit}$ by k_r, $\mathsf{K}_\mathrm{S}\ \mathsf{K}_\mathrm{R}\ \mathit{sbit}$ by k_{sr}, and $\mathsf{K}_\mathrm{R}\ \mathsf{K}_\mathrm{S}\ \mathsf{K}_\mathrm{R}\ \mathit{sbit}$ by k_{rsr}, the transition system Γ_{bt} is graphically given by



The states S_{bt} comprise of $\{\mathrm{z}_0, \mathrm{z}_1, \ldots, \mathrm{z}_7\}$ with $L_{bt}(\mathrm{z}_0) = \emptyset$, $L_{bt}(\mathrm{z}_1) = \{\mathrm{snt}\}$, $\ldots$, $L_{bt}(\mathrm{z}_7) = \{\mathrm{sbit}, \mathrm{rbit}, \mathrm{snt}, \mathrm{ack}\}$ as outlined in the graph above; the set of initial states is $S_{bt,0} = \{\mathrm{z}_0, \mathrm{z}_4\}$. The epistemic accessibility relations $E_{bt,a}$ for $a \in A_{bt}$ are given by *observability sets* $O_{bt,a}$ that declare two states $s_1, s_2 \in S_{bt}$ to be $O_{bt,a}$*-indistinguishable*, written as $s_1 \sim_{O_{bt,a}} s_2$, if for all $p \in O_{bt,a}$ it holds that $p \in L_{bt}(s_1) \iff p \in L_{bt}(s_2)$, and consequently $E_{bt,a} = \sim_{O_{bt,a}}$, such that $E_{bt,a}$ forms an equivalence relation. Due to $\mathrm{sbit} \notin O_{bt,\mathrm{R}}$, the receiver R cannot "see" sbit and hence cannot distinguish between states z_0 and z_4, but S can. On the other hand, R can distinguish between z_1 and z_5 as R has access to rbit. Finally, $\mathcal{T}_{bt}$ consists of two epistemically guarded actions

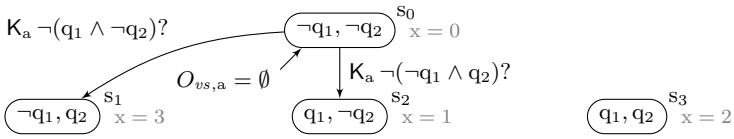
$$\neg\mathsf{K}_\mathrm{S}\ \mathsf{K}_\mathrm{R}\ \mathit{sbit} \supset \{(\mathrm{z}_i, \mathrm{z}_i) \mid 0 \leq i \leq 7\} \cup \{(\mathrm{z}_0, \mathrm{z}_1), (\mathrm{z}_2, \mathrm{z}_3), (\mathrm{z}_4, \mathrm{z}_5), (\mathrm{z}_6, \mathrm{z}_7)\} \text{ and}$$
$$\mathsf{K}_\mathrm{R}\ \mathit{sbit} \wedge \neg\mathsf{K}_\mathrm{R}\ \mathsf{K}_\mathrm{S}\ \mathsf{K}_\mathrm{R}\ \mathit{sbit} \supset \{(\mathrm{z}_i, \mathrm{z}_i) \mid 0 \leq i \leq 7\} \cup \{(\mathrm{z}_0, \mathrm{z}_2), (\mathrm{z}_1, \mathrm{z}_3), (\mathrm{z}_4, \mathrm{z}_6), (\mathrm{z}_5, \mathrm{z}_7)\},$$

which directly reflect the sending and acknowledging actions of the bit transmission problem: The system can only advance from z_0 to z_1 (and z_4 to z_5), where sending has been done successfully, if S does not know that R knows the bit; but it need not make such progress, i.e., sending can be unsuccessful. Similarly, the system can only advance from z_1 to z_3 (and z_5 to z_7), where an acknowledgement has been sent successfully, if R knows the bit and R does not know that S knows that R knows the bit.

(b) Consider the variable setting problem of the introduction for a single agent a:

$$\begin{array}{l}\textbf{if}\ \mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 1 \twoheadrightarrow \mathrm{x} \leftarrow 3 \\ \quad [\!]\ \mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 3 \twoheadrightarrow \mathrm{x} \leftarrow 1\ \textbf{fi}\end{array}$$

Encoding the integer $x \in \{0, 1, 2, 3\}$ by two bits q_1 and q_2, we model the problem as the following epistemically guarded transition system $\Gamma_{vs} = (\mathsf{B}_{vs}, \mathcal{T}_{vs})$ with $\mathsf{B}_{vs} = (S_{vs}, E_{vs}, L_{vs}, S_{vs,0})$ over $\Sigma_{vs} = (P_{vs}, A_{vs})$ with $P_{vs} = \{\mathrm{q}_1, \mathrm{q}_2\}$ and $A_{vs} = \{\mathrm{a}\}$:



$O_{vs,\mathrm{a}}$ represents a "blind" agent a that deems all states equally accessible. State s_3 is definitely not reachable. $\mathcal{T}_{vs}$ consists of the epistemically guarded actions

$$\mathsf{K}_\mathrm{a}\,\neg(\mathrm{q}_1 \wedge \neg\mathrm{q}_2) \supset \{(\mathrm{s}_0, \mathrm{s}_1)\} \quad \text{and} \quad \mathsf{K}_\mathrm{a}\,\neg(\neg\mathrm{q}_1 \wedge \mathrm{q}_2) \supset \{(\mathrm{s}_0, \mathrm{s}_2)\}\,. \qquad \square$$

3.2 Interpreting Epistemically Guarded Transition Systems

An epistemically guarded transition system $\Gamma = (S, E, L, S_0, \mathcal{T})$ over (P, A) is *interpreted* over an epistemic transition structure $M \in \mathscr{M}_{P,A}(S, E, L, S_0)$ by interpreting each guarded action $(\varphi \supset B) \in \mathcal{T}$ w.r.t. M as

$$(\varphi \supset B)^M = \{(s, s') \in B \mid s \in S_\omega(M) \text{ and } M, s \models \varphi\}\,,$$

and combining these interpretations into the epistemic transition structure

$$\Gamma^M = (S, E, L, S_0, \textstyle\bigcup_{\tau\in\mathcal{T}} \tau^M)\,.$$

We call M a *solution* for Γ if $\Gamma^M = M$.

Example 2. For the bit transmission problem as described in Ex. 1(a), the epistemic transition structure $M_{bt} = (\mathsf{B}_{bt}, T_{bt})$ with $T_{bt} = \{(\mathrm{z}_i, \mathrm{z}_i) \mid i \in \{0, 1, 3, 4, 5, 7\}\} \cup \{(\mathrm{z}_0, \mathrm{z}_1), (\mathrm{z}_1, \mathrm{z}_3), (\mathrm{z}_4, \mathrm{z}_5), (\mathrm{z}_5, \mathrm{z}_7)\}$ satisfies ${\Gamma_{bt}}^{M_{bt}} = M_{bt}$. This structure just omits the states z_2 and z_6 with $L_{bt}(\mathrm{z}_2) = \{\mathrm{ack}\}$ and $L_{bt}(\mathrm{z}_6) = \{\mathrm{sbit}, \mathrm{ack}\}$ which are definitely not reachable, as $\mathsf{K}_\mathrm{R}\, sbit$ is false in $\mathrm{z}_0 \sim_{O_{bt,\mathrm{R}}} \mathrm{z}_4$. Indeed,

$$\begin{aligned}
&M_{bt}, s \models \neg\mathsf{K}_\mathrm{S}\,\mathsf{K}_\mathrm{R}\, sbit \iff s \in \{\mathrm{z}_0, \mathrm{z}_1, \mathrm{z}_4, \mathrm{z}_5\}\\
&M_{bt}, s \models \mathsf{K}_\mathrm{R}\, sbit \iff s \in \{\mathrm{z}_1, \mathrm{z}_3, \mathrm{z}_5, \mathrm{z}_7\}\\
&M_{bt}, s \models \neg\mathsf{K}_\mathrm{R}\,\mathsf{K}_\mathrm{S}\,\mathsf{K}_\mathrm{R}\, sbit \iff s \in \{\mathrm{z}_0, \mathrm{z}_1, \mathrm{z}_3, \mathrm{z}_4, \mathrm{z}_5, \mathrm{z}_7\}
\end{aligned} \qquad \square$$

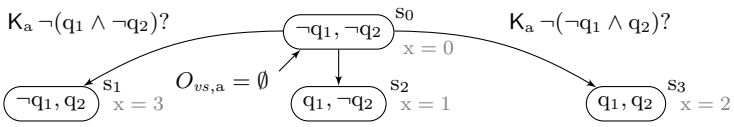
However, finding a solution is complicated by the fact that the functional of interpreting an epistemically guarded transition system over an epistemic transition structure is not monotone, in general, as illustrated by the following examples.

Example 3. (a) Continuing Ex. 1(b) for the variable setting problem Γ_{vs}, consider the epistemic transition structure $M_{vs,0} \in \mathscr{M}_{\Sigma_{vs}}(\mathsf{B}_{vs})$ with the empty transition relation $T(M_{vs,0}) = \emptyset$, and hence $S_0(M_{vs,0}) = \{\mathrm{s}_0\}$. Setting $M_{vs,i+1} = {\Gamma_{vs}}^{M_{vs,i}}$ for $0 \le i \le 2$ we obtain successively

τ	$\tau^{M_{vs,0}}$	$\tau^{M_{vs,1}}$	$\tau^{M_{vs,2}}$
$\mathsf{K}_\mathrm{a}\,\neg(\mathrm{q}_1 \wedge \neg\mathrm{q}_2) \supset \{(\mathrm{s}_0, \mathrm{s}_1)\}$	$\{(\mathrm{s}_0, \mathrm{s}_1)\}$	$\emptyset$	$\{(\mathrm{s}_0, \mathrm{s}_1)\}$
$\mathsf{K}_\mathrm{a}\,\neg(\neg\mathrm{q}_1 \wedge \mathrm{q}_2) \supset \{(\mathrm{s}_0, \mathrm{s}_2)\}$	$\{(\mathrm{s}_0, \mathrm{s}_2)\}$	$\emptyset$	$\{(\mathrm{s}_0, \mathrm{s}_2)\}$

In particular, $M_{vs,2} = {\Gamma_{vs}}^{M_{vs,1}} = {\Gamma_{vs}}^{{\Gamma_{vs}}^{M_{vs,0}}} = M_{vs,0}$. However, for $M_{vs,4}, M_{vs,5} \in \mathscr{M}_{\Sigma_{vs}}(\mathsf{B}_{vs})$ with $T(M_{vs,4}) = \{(\mathrm{s}_0, \mathrm{s}_1)\}$ and $T(M_{vs,5}) = \{(\mathrm{s}_0, \mathrm{s}_2)\}$ we obtain that ${\Gamma_{vs}}^{M_{vs,4}} = M_{vs,4}$ and ${\Gamma_{vs}}^{M_{vs,5}} = M_{vs,5}$.

(b) For capturing the cycle-breaking variable setting of the introduction consider the following epistemically guarded transition system $\Gamma_{vsb} = (\mathsf{B}_{vs}, \mathcal{T}_{vsb})$ over Σ_{vs} that shares the epistemic state basis B_{vs} with Ex. 1(b):



For $M_{vsb,0} = (\mathsf{B}_{vs}, \emptyset)$ with $S_0(M_{vsb,0}) = \{\mathrm{s}_0\}$, and setting $M_{vsb,i+1} = {\Gamma_{vsb}}^{M_{vsb,i}}$ for $0 \leq i \leq 3$ we obtain successively

τ	${\tau}^{M_{vsb,0}}$	${\tau}^{M_{vsb,1}}$	${\tau}^{M_{vsb,2}}$	${\tau}^{M_{vsb,3}}$
$\mathsf{K}_\mathrm{a}\,\neg(\mathrm{q}_1 \wedge \neg\mathrm{q}_2) \supset \{(\mathrm{s}_0, \mathrm{s}_1)\}$	$\{(\mathrm{s}_0, \mathrm{s}_1)\}$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathrm{true} \supset \{(\mathrm{s}_0, \mathrm{s}_2)\}$	$\{(\mathrm{s}_0, \mathrm{s}_2)\}$	$\{(\mathrm{s}_0, \mathrm{s}_2)\}$	$\{(\mathrm{s}_0, \mathrm{s}_2)\}$	$\{(\mathrm{s}_0, \mathrm{s}_2)\}$
$\mathsf{K}_\mathrm{a}\,\neg(\neg\mathrm{q}_1 \wedge \mathrm{q}_2) \supset \{(\mathrm{s}_0, \mathrm{s}_3)\}$	$\{(\mathrm{s}_0, \mathrm{s}_3)\}$	$\emptyset$	$\{(\mathrm{s}_0, \mathrm{s}_3)\}$	$\{(\mathrm{s}_0, \mathrm{s}_3)\}$

For $M_{vsb,3}$ with $S_\omega(M_{vsb,3}) = \{\mathrm{s}_0, \mathrm{s}_1, \mathrm{s}_3\}$ it finally holds that ${\Gamma_{vsb}}^{M_{vsb,3}} = M_{vsb,3}$. □

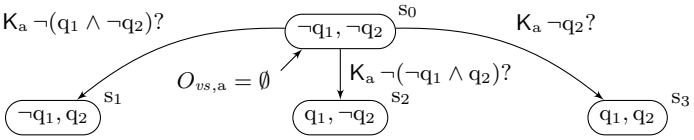
3.3 Iteration Semantics

For illustrating the non-monotonicity of the interpretation functional we have started the interpretation sequence for Γ with the smallest epistemic transition structure which suggests to look for a smallest fixed point — which need not exist. De Haan et al. [10] argue that a substitute consisting of the greatest fixed point would be more liberal. They construct a transfinite approximation sequence starting from an N_0 having all states reachable. For a successor ordinal $\alpha+1$, the approximation $N_{\alpha+1}$ is just the interpretation of Γ in N_α; for a limit ordinal λ, the approximation $N_\lambda = \bigcap_{\alpha<\lambda} \bigcup_{\alpha\leq\beta<\lambda} N_\beta$ is "the intersection of unions of approximations that are sufficiently close to the limit" [10, p. 269]. The latter is preferred over a union of intersections as it includes more states which implies less knowledge, such that "agents [know] facts only when there are good reasons for them" (ibid.). Due to cardinality reasons, the ordinal $\eta_\Gamma = \inf\{\alpha \mid \text{ex. } \beta \text{ s. t. } \alpha < \beta \text{ and } N_\alpha = N_\beta\}$ exists. If $N_{\alpha+1} \subseteq N_\alpha$ for all $\alpha \geq \eta_\Gamma$, then $N_{\eta_\Gamma+1} = N_{\eta_\Gamma}$; otherwise there is some $\alpha \geq \eta_\Gamma$ such that $N_{\alpha+1} \not\subseteq N_\alpha$. Thus $\alpha_\Gamma = \inf\{\alpha \mid \eta_\Gamma \leq \alpha \text{ and } (N_\alpha = N_{\alpha+1} \text{ or } N_{\alpha+1} \not\subseteq N_\alpha)\}$ exists and the *iteration semantics* of Γ is defined as N_{α_Γ}. This yields the greatest fixed point if the interpretation functional is monotone.

Example 4. (a) For the variable setting problem Γ_{vs} of Ex. 1(b) the interpretation sequence $(N_{vs,\alpha})_{0\leq\alpha}$ starts with $N_{vs,0}$ showing $T(N_{vs,0}) = S_{vs} \times S_{vs}$. Using the epistemic transition structures from Ex. 3(a) it holds that $N_{vs,k+1} = {\Gamma_{vs}}^{N_{vs,k}} = M_{vs,2}$ for k even and $N_{vs,k+1} = M_{vs,1}$ for $k \geq 1$ odd. Thus, $N_{vs,1} = N_{vs,3}$ such that $\eta_{\Gamma_{vs}} = 1 = \alpha_{\Gamma_{vs}}$, since $T(N_{vs,2}) = \{(\mathrm{s}_0, \mathrm{s}_0), (\mathrm{s}_0, \mathrm{s}_1), (\mathrm{s}_0, \mathrm{s}_2)\} \not\subseteq \emptyset = T(N_{vs,1})$. Hence the iteration semantics of Γ_{vs} is given by $N_{vs,1} = M_{vs,2}$; since its transition relation is empty, Γ_{vs} has the same iteration semantics as an epistemically guarded transition system without any guarded actions.

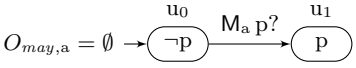
(b) Computing the iteration semantics sequence $(N_{vsb,\alpha})_{0\leq k}$ of the cycle-breaking variable setting Γ_{vsb} of Ex. 3(b) proceeds as $N_{vsb,k} = M_{vsb,k+1}$. Since this time the functional is monotone from $\alpha = 1$ onwards, the iteration semantics is $N_{vsb,2}$.

(c) Consider the following epistemically guarded transition system $\Gamma_{nc} = (\mathsf{B}_{vs}, \mathcal{T}_{nc})$ over Σ_{vs} that shares the epistemic basis B_{vs} with the variable setting problem Γ_{vs} of (a) and only adds the guarded action $\mathsf{K}_\mathrm{a}\,\neg\mathrm{q}_2 \supset \{(\mathrm{s}_0, \mathrm{s}_3)\}$:



The interpretation process runs as for Γ_{vs}, and the epistemic transition structure with the empty transition relation is also the iteration semantics of Γ_{nc}. This time, however, there is a unique non-empty interpretation, viz. the transition structure consisting only of $(\mathrm{s}_0, \mathrm{s}_1)$. Finding this solution is not constructive and some speculation is necessary: there is no solution where s_2 is reachable; if s_2 were reachable, then s_1 would be reachable leading to a contradiction due to the (non-)reachability of s_3. Thus only the possibility of s_0 and s_1 being reachable, and s_2 and s_3 unreachable, remains.

(d) For the epistemically guarded transition system Γ_{may} over $(\{\mathrm{p}\}, \{\mathrm{a}\})$ given by



the iteration process when started with $N_{may,0}$ having $T(N_{may,0}) = \{\mathrm{u}_0, \mathrm{u}_1\} \times \{\mathrm{u}_0, \mathrm{u}_1\}$ evaluates $\mathsf{M}_\mathrm{a}\,\mathrm{p}$ to true and we obtain $N_{may,1}$ with $T(N_{may,1}) = \{(\mathrm{u}_0, \mathrm{u}_1)\}$ which in turn is confirmed by the next iteration yielding a fixed point. This iteration semantics, however, has a touch of a "vaticinium ex eventu": p can be reached since p may be reached. □

3.4 Unique Interpretation Solutions

A knowledge-based program can be executed reliably just step by step if each knowledge guard can be stably decided based on what has been computed up to the current point of execution. In particular, in order to obtain a solution by execution, knowledge must not be invalidated by information only to be gained later on. Conversely, if all knowledge guards can be decided by just looking to the past, there is at most a single solution.

Based on this observation, Fagin et al. [13,14] develop a formal characterisation of unique interpretability by capturing the notion that solutions "depend on the past". They then show that "providing epistemic witnesses" is a sufficient criterion for "dependence on the past", which in turn always holds for "synchronous" programs. We briefly summarise their main line of argument adapting the demonstration from their run-based account for knowledge-based programs to our state-based epistemically guarded transition systems.[3]

[3] The proofs are available in a long version at https://arxiv.org/abs/2301.10807.

An epistemic formula $\varphi \in \Phi_{P,A}$ is said to *depend on the past* w. r. t. a class of epistemic transition structures $\mathcal{M} \subseteq \mathscr{M}_{P,A}(\mathsf{B})$ if for all $M_1, M_2 \in \mathcal{M}$ and all $k \in \mathbb{N}$ it holds that $T_k(M_1) = T_k(M_2)$ implies $M_1, s \models \varphi \iff M_2, s \models \varphi$ for all $s \in S_k(M_1) \cap S_k(M_2)$; an epistemically guarded transition system $\Gamma = (\mathsf{B}, \mathcal{T})$ over (P, A) is *depending on the past* w. r. t. $\mathcal{M}$ if every φ in $(\varphi \supset B) \in \mathcal{T}$ depends on the past w. r. t. $\mathcal{M}$.

Example 5. For Ex. 3(a) neither $\mathsf{K}_\mathrm{a} \neg(\mathrm{q}_1 \wedge \neg \mathrm{q}_2)$ nor $\mathsf{K}_\mathrm{a} \neg(\neg \mathrm{q}_1 \wedge \mathrm{q}_2)$ depends on the past w. r. t. $\{M_{vs,0}, M_{vs,1}\}$. In particular, $T_0(M_{vs,0}) = \emptyset = T_0(M_{vs,1})$ and $S_0(M_{vs,0}) = \{\mathrm{s}_0\} = S_0(M_{vs,1})$, but $M_{vs,0}, \mathrm{s}_0 \models \mathsf{K}_\mathrm{a} \neg(\mathrm{q}_1 \wedge \neg \mathrm{q}_2)$ and $M_{vs,1}, \mathrm{s}_0 \not\models \mathsf{K}_\mathrm{a} \neg(\mathrm{q}_1 \wedge \neg \mathrm{q}_2)$. Similarly for Ex. 3(b), these two formulæ do not depend on the past w. r. t. $\{M_{vsb,0}, M_{vsb,1}, M_{vsb,2}, M_{vsb,3}\}$, but they do w. r. t. $\{M_{vsb,1}, M_{vsb,2}, M_{vsb,3}\}$. □

An epistemically guarded transition system Γ has at most one solution if, and only if, it depends on the past w. r. t. all its solutions. Due to the dependence on the past the successive reachable transition relations $T_k(M)$ of all solutions $M = \Gamma^M$, i. e., their pasts, coincide.

Proposition 1. *Let $\Gamma = (\mathsf{B}, \mathcal{T})$ be an epistemically guarded transition system over Σ. Then Γ has at most one solution if, and only if, there is an $\mathcal{M} \subseteq \mathscr{M}_\Sigma(\mathsf{B})$ with $\{M \in \mathscr{M}_\Sigma(\mathsf{B}) \mid \Gamma^M = M\} \subseteq \mathcal{M}$ such that Γ depends on the past w. r. t. $\mathcal{M}$.*

In order to obtain a solution of Γ by execution, the system is interpreted repeatedly to construct the approximations $(M_k)_{0 \leq k}$ with $M_{k+1} = \Gamma^{M_k}$ for $k \geq -1$ starting with some M_{-1}. Each approximation M_k with $k \geq 0$ contributes a transition relation $T_k(M_k)$ which can be combined into a limit M_ω. If Γ depends on the past w. r. t. the class of epistemic transition structures from which the approximands are constructed and which also contains the limit, then the interpretation of the limit M_ω yields a fixed point.

Proposition 2. *Let $\Gamma = (\mathsf{B}, \mathcal{T})$ be an epistemically guarded transition system over Σ, let $\mathcal{M} \subseteq \mathscr{M}_\Sigma(\mathsf{B})$ such that $\Gamma^M \in \mathcal{M}$ for every $M \in \mathcal{M}$ and $(\mathsf{B}, \bigcup_{0 \leq k} T_k(M_k)) \in \mathcal{M}$ for all $(M_k)_{0 \leq k} \subseteq \mathcal{M}$ with $T_k(M_{k'}) = T_k(M_k)$ for all $k' \geq k \geq 0$, and let Γ depend on the past w. r. t. $\mathcal{M}$. Let $M_{-1} \in \mathcal{M}$, $M_{i+1} = \Gamma^{M_i}$ for all $i \geq -1$, and $M_\omega = (\mathsf{B}, \bigcup_{0 \leq k} T_k(M_k))$. Then $\Gamma^{M_\omega} = \Gamma^{\Gamma^{M_\omega}}$.*

A sufficient criterion for obtaining a comprehensive class of epistemic transition structures $\mathcal{M}$ such that Γ depends on the past w. r. t. $\mathcal{M}$ is provided by epistemic witnesses: If some knowledge formula $\mathsf{K}_a \varphi$ of Γ does not hold at some state of an interpreting epistemic transition structure there is evidence in the past of this structure why it does not hold. Formally, a structure $M \in \mathscr{M}_{P,A}(\mathsf{B})$ *provides epistemic witnesses* for a formula $\mathsf{K}_a \varphi \in \Phi_{P,A}$ if for all $k \geq 0$, $s \in S_k(M)$ it holds that if $M, s \not\models \mathsf{K}_a \varphi$, then there is an $s' \in S_k(M)$ with $(s, s') \in E_a$ and $M, s' \not\models \varphi$.

Lemma 1. *Let $\Gamma = (\mathsf{B}, \mathcal{T})$ be an epistemically guarded transition system over Σ and let $\mathcal{M} \subseteq \mathscr{M}_\Sigma(\mathsf{B})$ such that all $M \in \mathcal{M}$ provide epistemic witnesses for all knowledge guards in Γ. Then Γ is depending on the past w. r. t. $\mathcal{M}$.*

A sufficient criterion, in turn, for a structure $M \in \mathscr{M}_{P,A}(S, E, L, S_0)$ to provide epistemic witnesses is M being *synchronous*: if for all $a \in A$ and all reachable $s_1 \in$

$S_{k_1}(M)$ and $s_2 \in S_{k_2}(M)$ with $(s_1, s_2) \in E_a$ it holds that $s_1, s_2 \in S_{\min\{k_1,k_2\}}(M)$. In a synchronous structure the temporal and the epistemic dimension for each agent are hence tightly coupled and agents cannot access the future, but also do not need to know the future.

Example 6. The interpretation M_{bt} of the bit transmission problem given in Ex. 2 provides epistemic witnesses, but is not synchronous: the sender S cannot distinguish z_0 reachable at depth 0 of M_{bt} from z_1 that is only reachable at depth 1, and similarly the receiver R cannot distinguish z_1 from z_3 at the respective depths of 1 and 2. □

An epistemically guarded transition system $\Gamma = (\mathsf{B}, \mathcal{T})$ over Σ *provides epistemic witnesses* if for each $M \in \mathscr{M}_\Sigma(\mathsf{B})$ the interpretation Γ^M provides epistemic witnesses for all knowledge formulæ occurring in some of the action guards of Γ; Γ is *synchronous* if each Γ^M is synchronous. Moreover, Γ can syntactically be seen to be synchronous (cf. [14, p. 135]) if it is round-based where all agents perform some action in each round and record locally which actions they have taken.

4 (Re-)Interpreting Knowledge-based Programs

The results by Fagin et al. [13,14] guarantee a unique interpretation for all synchronous knowledge-based programs; the approach by De Haan et al. [10] aims at extending the interpretation to asynchronous programs, but assigns semantics also to contradictory or self-fulfilling programs.

The necessity of avoiding contradictory or self-fulfilling behaviour already occurs in the design of synchronous programming languages [6]: Their underlying principle is "perfect synchrony", that any reaction of a program takes zero time and that thus whatever is output in reaction to some input is already present at the same time as the input. Since the presence or absence of signals can be tested, this requires "logical coherence" saying that a (non-input) signal is present in a reaction if, and only if, this signal is emitted in this very reaction. A program needs to be both *reactive* in the sense of leading to some logically coherent signal status, and *determinate*, i.e., not showing several such statuses. For example, in Esterel [7], the program fragment

```
present S then nothing else emit S end
```

is not reactive, but contradictory: signal S is only emitted if it is not emitted; and

```
present S then emit S else nothing end
```

is not determinate, but self-fulfilling: S is emitted if it is emitted, and it is not emitted if it is not. Such programs can be revealed by using a cycle-detecting static analysis, as is done in Lustre [18], or, for including more intricate cases, by Berry's "constructive semantics" as for Esterel [8]. Building on a "logical semantics" recording what is emitted in each step of execution, a *must*/*cannot* analysis is performed: what must/cannot be emitted, which branch must/cannot be executed. It is then required that for each signal it can be decided whether it must be present or it cannot be present. For example, in the parallel execution

```
   [ present S1 then emit S1 end ]
|| [ present S1 then present S2 then nothing else emit S2 end end ]
```

both signals can be emitted — if S1 is assumed to be present, and S2 absent —, but none must be emitted. Thus the constructive semantics does not reach a decision of what must/cannot be present and the program is not constructive. Intriguingly, however, there is exactly one coherent signal status that can be reached by execution: S1 and S2 absent.

We adapt Berry's constructive semantics approach to knowledge-based programs. In fact, the first, non-reactive Esterel program fragment resembles the variable setting problem described in Ex. 3(a), the second, non-determinate fragment directly corresponds to Ex. 4(d), and the last, combined fragment is essentially the same as Ex. 4(c). We first define a must/can version of epistemic transition structures with a lower (must) and an upper bound (can). Based on a positive (must) and negative (cannot) satisfaction relation of epistemic formulæ over these structures we show how an epistemically guarded transition system can be interpreted yielding another epistemic must/can transition structure. For uniformity, we rephrase this interpretation in terms of the negation normal form of formulæ and demonstrate that the constructive interpretation is always monotone and leads to a least fixed point. For any knowledge-based program, this fixed point soundly shows which executions are necessary and which are possible. However, the fixed point need not be decided, and more can be possible than is necessary. We show that synchronous programs always lead to decided fixed points.

4.1 Epistemic Must/Can Transition Structures

An *epistemic must/can transition structure* $Y = (S, E, L, S_0, (T_\mu, T_\nu))$ over $\Sigma = (P, A)$ is given by an epistemic state basis $\mathsf{B} = (S, E, L, S_0)$ and two *lower* and *upper transition relations* $T_\mu, T_\nu \subseteq S \times S$ with $T_\mu \subseteq T_\nu$. In particular, $Y_\mu = (\mathsf{B}, T_\mu)$ and $Y_\nu = (\mathsf{B}, T_\nu)$ are epistemic transition structures over Σ with $Y_\mu \subseteq Y_\nu$.

The *positive* and *negative satisfaction relations* of an epistemic formula $\varphi \in \Phi_{P,A}$ over the epistemic must/can transition structure Y at a state $s \in S_\omega(Y_\nu)$, written $Y, s \models_\mathrm{p} \varphi$ and $Y, s \models_\mathrm{n} \varphi$, are defined as follows:

$$
\begin{array}{ll}
Y, s \models_\mathrm{p} p \iff p \in L(s) & Y, s \models_\mathrm{n} p \iff p \notin L(s) \\
Y, s \not\models_\mathrm{p} \mathrm{false} & Y, s \models_\mathrm{n} \mathrm{false} \\
Y, s \models_\mathrm{p} \neg\varphi \iff Y, s \models_\mathrm{n} \varphi & Y, s \models_\mathrm{n} \neg\varphi \iff Y, s \models_\mathrm{p} \varphi \\
Y, s \models_\mathrm{p} \varphi_1 \wedge \varphi_2 \iff & Y, s \models_\mathrm{n} \varphi_1 \wedge \varphi_2 \iff \\
\quad Y, s \models_\mathrm{p} \varphi_1 \text{ and } Y, s \models_\mathrm{p} \varphi_2 & \quad Y, s \models_\mathrm{n} \varphi_1 \text{ or } Y, s \models_\mathrm{n} \varphi_2 \\
Y, s \models_\mathrm{p} \mathsf{K}_a \varphi \iff Y, s' \models_\mathrm{p} \varphi & Y, s \models_\mathrm{n} \mathsf{K}_a \varphi \iff Y, s' \models_\mathrm{n} \varphi \\
\quad \text{for all } s' \in S_\omega(Y_\nu) & \quad \text{for some } s' \in S_\omega(Y_\mu) \\
\quad \text{with } (s, s') \in E_a & \quad \text{with } (s, s') \in E_a
\end{array}
$$

A formula is positively satisfied over Y if it must be true given the upper bound Y_ν of possible behaviour, it is negatively satisfied if it cannot be true given the lower bound Y_μ of necessary behaviour. In fact, it holds that what must be true can also be true:[4]

Lemma 2. *Let $Y = (S, E, L, S_0, (T_\mu, T_\nu))$ be an epistemic must/can transition structure over (P, A) and $\varphi \in \Phi_{P,A}$. Then for all $s \in S_\omega(Y_\nu)$, $Y, s \models_\mathrm{p} \varphi$ implies $Y, s \not\models_\mathrm{n} \varphi$.*

[4] The proofs are available in a long version at https://arxiv.org/abs/2301.10807.

The set of epistemic must/can transition structures over Σ and the epistemic state basis B is denoted by $\mathscr{Y}_\Sigma(\mathsf{B})$. We say that $Y_1 \sqsubseteq Y_2$ for $Y_1, Y_2 \in \mathscr{Y}_\Sigma(\mathsf{B})$ if $Y_{1,\mu} \subseteq Y_{2,\mu}$ and $Y_{1,\nu} \supseteq Y_{2,\nu}$: an *extension* raises the lower bound and reduces the upper bound.

As with epistemic transition structures, an epistemically guarded transition system $\Gamma = (S, E, L, S_0, \mathcal{T})$ over (P, A) can be interpreted over an epistemic must/can transition structure $Y \in \mathscr{Y}_{P,A}(S, E, L, S_0)$: The *interpretation* of a guarded action $(\varphi \supset B) \in \mathcal{T}$ w.r.t. to Y is given by the pair $(\varphi \supset B)^Y = ((\varphi \supset B)^{Y,\mu}, (\varphi \supset B)^{Y,\nu})$ with

$$(\varphi \supset B)^{Y,\mu} = \{(s, s') \in B \mid s \in S_\omega(Y_\mu) \text{ and } Y, s \models_{\mathrm{p}} \varphi\} ,$$
$$(\varphi \supset B)^{Y,\nu} = \{(s, s') \in B \mid s \in S_\omega(Y_\nu) \text{ and } Y, s \not\models_{\mathrm{n}} \varphi\} .$$

By Lem. 2 it holds that $\tau^{Y,\mu} \subseteq \tau^{Y,\nu}$ for each $\tau \in \mathcal{T}$. The *constructive interpretation* of Γ w.r.t. Y is given by the epistemic must/can transition structure

$$\Gamma^Y = (S, E, L, S_0, (\textstyle\bigcup_{\tau\in\mathcal{T}} \tau^{Y,\mu}, \bigcup_{\tau\in\mathcal{T}} \tau^{Y,\nu})) .$$

This is well defined, i.e., $(\Gamma^Y)_\mu \subseteq (\Gamma^Y)_\nu$. We call Y a *constructive solution* for Γ if $\Gamma^Y = Y$; a constructive solution is *decided* if $Y_\mu = Y_\nu$.

Again as with epistemic transition structures, this interpretation over epistemic must/can transition structures can be iterated for finally reaching a stable structure — and this time interpretation turns out to be monotone.

Example 7. (a) Re-consider the cycle-breaking variable setting problem of Ex. 3(b). We start the interpretation in $Y_{vsb,0} = (\mathsf{B}_{vs}, (\emptyset, S_{vs}^2))$ and successively obtain the following epistemic must/can transition structures:

τ	$\tau^{Y_{vsb,0}}$	$\tau^{Y_{vsb,1}}$	$\tau^{Y_{vsb,2}}$	$\tau^{Y_{vsb,3}}$
$\mathsf{K}_{\mathrm{a}}\,\neg(\mathrm{q}_1 \wedge \neg\mathrm{q}_2) \supset \{(\mathrm{s}_0,\mathrm{s}_1)\}$	$\emptyset$ $\{(\mathrm{s}_0,\mathrm{s}_1)\}$	$\emptyset$ $\emptyset$	$\emptyset$ $\emptyset$	$\emptyset$ $\emptyset$
$\mathrm{true} \supset \{(\mathrm{s}_0,\mathrm{s}_2)\}$	$\{(\mathrm{s}_0,\mathrm{s}_2)\}$ $\{(\mathrm{s}_0,\mathrm{s}_2)\}$	$\{(\mathrm{s}_0,\mathrm{s}_2)\}$ $\{(\mathrm{s}_0,\mathrm{s}_2)\}$	$\{(\mathrm{s}_0,\mathrm{s}_2)\}$ $\{(\mathrm{s}_0,\mathrm{s}_2)\}$	$\{(\mathrm{s}_0,\mathrm{s}_2)\}$ $\{(\mathrm{s}_0,\mathrm{s}_2)\}$
$\mathsf{K}_{\mathrm{a}}\,\neg(\neg\mathrm{q}_1 \wedge \mathrm{q}_2) \supset \{(\mathrm{s}_0,\mathrm{s}_3)\}$	$\emptyset$ $\{(\mathrm{s}_0,\mathrm{s}_3)\}$	$\emptyset$ $\{(\mathrm{s}_0,\mathrm{s}_3)\}$	$\{(\mathrm{s}_0,\mathrm{s}_3)\}$ $\{(\mathrm{s}_0,\mathrm{s}_3)\}$	$\{(\mathrm{s}_0,\mathrm{s}_3)\}$ $\{(\mathrm{s}_0,\mathrm{s}_3)\}$

Not only does it hold that ${\Gamma_{vsb}}^{Y_{vsb,3}} = Y_{vsb,3}$, but the interpretations indeed evolve monotonically w.r.t. $\sqsubseteq$. Moreover, the structure $Y_{vsb,3}$ is decided and everything what can happen also must happen, i.e., $(Y_{vsb,3})_\mu = (Y_{vsb,3})_\nu$.

(b) For the cyclic variable setting problem, see Ex. 1(b) and Ex. 3(a), the interpretation process is monotone, but only yields

τ	$\tau^{Y_{vs,0}}$	$\tau^{Y_{vs,1}}$
$\mathsf{K}_{\mathrm{a}}\,\neg(\mathrm{q}_1 \wedge \neg\mathrm{q}_2) \supset \{(\mathrm{s}_0,\mathrm{s}_1)\}$	$(\emptyset, \{(\mathrm{s}_0,\mathrm{s}_1)\})$	$(\emptyset, \{(\mathrm{s}_0,\mathrm{s}_1)\})$
$\mathsf{K}_{\mathrm{a}}\,\neg(\neg\mathrm{q}_1 \wedge \mathrm{q}_2) \supset \{(\mathrm{s}_0,\mathrm{s}_2)\}$	$(\emptyset, \{(\mathrm{s}_0,\mathrm{s}_2)\})$	$(\emptyset, \{(\mathrm{s}_0,\mathrm{s}_2)\})$

The epistemic must/can transition structure $Y_{vs,1}$ is not decided, and indeed there are two solutions of Γ_{vs} in terms of epistemic transition structures. However, the same undecidedness holds true for Γ_{nc} of Ex. 4(c), that is, the unique solution is also missed by the constructive interpretation. □

4.2 Constructive Interpretation

The separated positive (must) and negative (cannot) satisfaction relations over an epistemic must/can transition structure $Y \in \mathscr{Y}_{P,A}(S, E, L, S_0)$ can be merged into a single, uniform satisfaction relation relying on the *negation normal form* of epistemic formulæ where negation only occurs in front of propositions. For an arbitrary $\varphi \in \Phi_{P,A}$ there exists an equivalent $\mathrm{nnf}(\varphi) \in \Phi_{P,A}$ in negation normal form, such that, in particular

$$\mathrm{nnf}(\neg p) = \neg p \qquad \mathrm{nnf}(\neg\neg\varphi) = \mathrm{nnf}(\varphi)$$
$$\mathrm{nnf}(\neg\mathrm{false}) = \mathrm{true} \qquad \mathrm{nnf}(\neg(\varphi_1 \wedge \varphi_2)) = \mathrm{nnf}(\neg\varphi_1) \vee \mathrm{nnf}(\neg\varphi_2)$$
$$\mathrm{nnf}(\neg \mathsf{K}_a\, \varphi) = \mathsf{M}_a\, \mathrm{nnf}(\neg\varphi)$$

The *constructive satisfaction relation* $Y, s \models \varphi$ for a state $s \in S_\omega(Y_\nu)$ and an epistemic formula $\varphi \in \Phi_{P,A}$ in negation normal form is defined just as for arbitrary epistemic formulæ, but using the upper bound Y_ν for the universal quantifier of K_a and the lower bound Y_μ for the existential quantifier of M_a; in particular,

$$Y, s \models \neg p \iff p \notin L(s)$$
$$Y, s \models \mathsf{K}_a\, \varphi \iff Y, s' \models \varphi \text{ f. a. } s' \in S_\omega(Y_\nu) \text{ with } (s, s') \in E_a$$
$$Y, s \models \mathsf{M}_a\, \varphi \iff \text{ex. } s' \in S_\omega(Y_\mu) \text{ s. t. } (s, s') \in E_a \text{ and } Y, s' \models \varphi$$

The constructive satisfaction relation indeed combines $\models_{\mathrm{p}}$ and $\models_{\mathrm{n}}$:

Lemma 3. *Let $Y \in \mathscr{Y}_{P,A}(\mathsf{B})$, $\varphi \in \Phi_{P,A}$, and $s \in S_\omega(Y_\nu)$. Then $Y, s \models_{\mathrm{p}} \varphi$ iff $Y, s \models \mathrm{nnf}(\varphi)$ and $Y, s \models_{\mathrm{n}} \varphi$ iff $Y, s \models \mathrm{nnf}(\neg\varphi)$.*

It follows that if $Y_\mu = Y_\nu$, then $Y, s \models \varphi$ if, and only if, $Y_\mu, s \models \varphi$ or, equivalently, $Y_\nu, s \models \varphi$. We also obtain that constructive satisfaction is preserved when extending epistemic must/can transition structures:

Lemma 4. *Let $Y, Y' \in \mathscr{Y}_{P,A}(\mathsf{B})$ with $Y \sqsubseteq Y'$ and let $\varphi \in \Phi_{P,A}$. Then $Y, s \models \mathrm{nnf}(\varphi)$ implies $Y', s \models \mathrm{nnf}(\varphi)$ for all $s \in S_\omega(Y'_\nu)$.*

This preservation of satisfaction yields that constructive interpretation is monotone.

Proposition 3. *Let $\Gamma = (\mathsf{B}, \mathcal{T})$ be an epistemically guarded transition system over Σ and $Y, Y' \in \mathscr{Y}_\Sigma(\mathsf{B})$ such that $Y \sqsubseteq Y'$. Then $\Gamma^Y \sqsubseteq \Gamma^{Y'}$.*

Finally, we can observe that $\mathscr{Y}_\Sigma(\mathsf{B})$ for $\mathsf{B} = (S, E, L, S_0)$ with the ordering $\sqsubseteq$ is an *inductive partial order*: each directed subset $\Delta \subseteq \mathscr{Y}_\Sigma(\mathsf{B})$ has a least upper bound $\bigsqcup \Delta$ w. r. t. $\sqsubseteq$, where *directed* means that every two $Y_1, Y_2 \in \Delta$ have an upper bound $Y \in \Delta$ such that $Y_1 \sqsubseteq Y$ and $Y_2 \sqsubseteq Y$; and there is also a *bottom* or least element $\bot_{\Sigma,\mathsf{B}} = (S, E, L, S_0, (\emptyset, S \times S)) \in \mathscr{Y}_\Sigma(\mathsf{B})$.

Proposition 4. *$(\mathscr{Y}_\Sigma(\mathsf{B}), \sqsubseteq, \bot_{\Sigma,\mathsf{B}})$ is an inductive partial order.*

Pataraia's fixed-point theorem [9, §8.22] now guarantees that the monotone operator $Y \mapsto \Gamma^Y$ for each epistemically guarded transition system $\Gamma = (\mathsf{B}, \mathcal{T})$ has a least fixed point in the inductive partial order. It can be computed by, possibly transfinite, iterated application of constructive interpretation to $\bot_{\Sigma,\mathsf{B}}$, that is, $Y_0 = \bot_{\Sigma,\mathsf{B}}$, $Y_{\alpha+1} = \Gamma^{Y_\alpha}$ for a successor ordinal $\alpha + 1$, and $Y_\lambda = \bigsqcup_{\alpha<\lambda} Y_\alpha$ until equality [9, Exc. 8.19]. Compared to the iteration semantics of Sect. 3.3, the computation of the constructive semantics thus does not have to record all previous approximations in order to find a repetition.

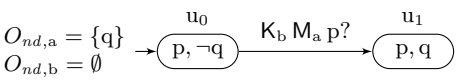
4.3 (Un-)Decided Constructive Fixed Points

If any constructive fixed point $Y = \Gamma^Y$ with $Y \in \mathscr{Y}_\Sigma(\mathsf{B})$ is decided, then there is the solution $Y_\mu = \Gamma^{Y_\mu} = \Gamma^{Y_\nu} = Y_\nu$ in terms of epistemic transition structures, and Γ is not contradictory. Even if it is not decided, the must/can structures $Y_{\mu\mu} = (\mathsf{B}, (T(Y_\mu), T(Y_\mu))) \in \mathscr{Y}_\Sigma(\mathsf{B})$ and $Y_{\nu\nu} = (\mathsf{B}, (T(Y_\nu), T(Y_\nu))) \in \mathscr{Y}_\Sigma(\mathsf{B})$ satisfy $Y \sqsubseteq Y_{\mu\mu}$ and $Y \sqsubseteq Y_{\nu\nu}$, such that by Prop. 3 we obtain $Y = \Gamma^Y \sqsubseteq \Gamma^{Y_{\mu\mu}}, \Gamma^{Y_{\nu\nu}}$ which yields $Y_\mu \subseteq \Gamma^{Y_\mu}$ and $\Gamma^{Y_\nu} \subseteq Y_\nu$, but not equality, in general. For the least constructive fixed point $\mu\Gamma$, any solution $M = \Gamma^M$ thus satisfies $(\mu\Gamma)_\mu \subseteq M \subseteq (\mu\Gamma)_\nu$, always giving sound lower and upper bounds and, if $\mu\Gamma$ is decided, moreover unique solvability:

Proposition 5. *Let $\Gamma = (\mathsf{B}, \mathcal{T})$ be an epistemically guarded transition system over Σ and assume $\mu\Gamma \in \mathscr{Y}_\Sigma(\mathsf{B})$ is decided. Then Γ has a unique solution in $\mathscr{M}_\Sigma(\mathsf{B})$.*

Still, even for epistemically guarded transition systems that provide epistemic witnesses it is not guaranteed that the least constructive fixed point is decided:

Example 8. Consider the following epistemically guarded transition system $\Gamma_{nd} = (\mathsf{B}_{nd}, \mathcal{T}_{nd})$ over $\Sigma_{nd} = (P_{nd}, A_{nd})$ with $P_{nd} = \{\mathrm{p}, \mathrm{q}\}$ and $A_{nd} = \{\mathrm{a}, \mathrm{b}\}$:



Constructive interpretation yields the non-decided fixed point Y_{nd} with $T(Y_{nd,\mu}) = \emptyset$ and $T(Y_{nd,\nu}) = \{(\mathrm{u}_0, \mathrm{u}_1)\}$, as $Y_{nd}, \mathrm{u}_0 \not\models \mathsf{K}_\mathrm{b}\,\mathsf{M}_\mathrm{a}\,\mathrm{p}$, but also $Y_{nd}, \mathrm{u}_0 \not\models \mathsf{M}_\mathrm{b}\,\mathsf{K}_\mathrm{a}\,\neg\mathrm{p}$: the states u_0 and u_1 can be distinguished by agent a, and agent b cannot tell whether a step has been taken. In u_0 the formula $\mathsf{M}_\mathrm{a}\,\mathrm{p}$ holds w.r.t. Y_{nd}, but in u_1 it does not, since $(\mathrm{u}_1, \mathrm{u}_0) \notin E_{nd,a}$. On the other hand, Γ_{nd} provides epistemic witnesses pathologically, since ${\Gamma_{nd}}^M, s \models \mathsf{K}_\mathrm{b}\,\mathsf{M}_\mathrm{a}\,\mathrm{p}$ for any $M \in \mathscr{M}_{\Sigma_{nd}}(\mathsf{B}_{nd})$ and any $s \in S_\omega({\Gamma_{nd}}^M)$, and hence has a unique interpretation, which in this case is ${\Gamma_{nd}}^{Y_{nd,\mu}} = Y_{nd,\nu} = {\Gamma_{nd}}^{Y_{nd,\nu}}$. □

For synchronous epistemically guarded transition systems, however, the least fixed point is decided, since all knowledge refers to a past that must have happened:

Lemma 5. *Let $\Gamma = (\mathsf{B}, \mathcal{T})$ be an epistemically guarded transition system over Σ that is synchronous. Let $Y \in \mathscr{Y}_\Sigma(\mathsf{B})$ satisfy $\Gamma^Y = Y$. Then Y is decided.*

Summing up, the constructive approach to interpreting knowledge-based programs subsumes the solutions for synchronous programs and provides a sound procedure for obtaining lower and upper bounds for the execution of both synchronous and asynchronous programs. The approach, however, is not complete: If the least constructive fixed point $\mu\Gamma$ is undecided, a system Γ may be contradictory without any solution (see Ex. 3(a)), self-fulfilling with several solutions (see Ex. 4(d)), or it may have a unique solution in terms of epistemic transition structures (see Ex. 4(c)). One strategy that suggests itself for analysing Γ further is to check whether an interpretation using the lower bound $(\mu\Gamma)_\mu$ of the least fixed point satisfies $\Gamma^{(\mu\Gamma)_\mu} = (\mu\Gamma)_\nu = \Gamma^{(\mu\Gamma)_\nu}$, which means that when executing according to what must happen all what can happen is already covered (see Ex. 8).

5 Knowledge-based Programs as Rule Systems

The "executions" of an epistemically guarded transition system Γ can be captured as derivations of two mutually dependent inductive rule systems, like used for inductive definitions [1,19]. One rule system defines the reachability in Γ, the other one the satisfaction of knowledge formulæ in negation normal form over Γ. When Γ provides epistemic witnesses, the mutual dependence can be resolved by stratifying the rule system for reachability according to the depth of the execution. In the general case, the non-monotone dependence of the formula satisfaction system on the reachability system — the more states are reachable, the less is known — can be mitigated by extending the notion of rule systems to include also negative premisses: The conclusion of a rule is derivable if all its (positive) premisses are derivable, but none of its negative premisses. When applied to knowledge formulæ, negative premisses express that no counterexample is reachable.

The general rule systems can also be read as logic programs with "negation as failure" [11]. A direct application of the must/can approximation technique to the general rule system or, equivalently, the logic program resulting from a knowledge-based program reconstructs the Kripke-Kleene fixed point; the possible solutions correspond to "stable models" [16].

5.1 Inductive Rule Systems

An *inductive rule system* R consists of *rules* of the form X/y where the *premisses* $X \subseteq U$ and the *conclusion* $y \in U$ are drawn from some *universe* of *judgements* U. A rule X/y is interpreted as "if all X can be inferred, then y can be inferred". The *derivations* in R together with their *sets of premisses* and *conclusions* are inductively defined as follows:

- a $y \in U$ is itself a derivation; its set of premisses is $\{y\}$, its conclusion is y;
- if $X/y \in R$ and $(d_x)_{x\in X}$ a family of derivations with conclusions $(x)_{x\in X}$, then $(d_x)_{x\in X}/y$ is a derivation; its set of premisses is the union of the premisses of $(d_x)_{x\in X}$, its conclusion is y.

A $y \in U$ is *derivable* in R if there is a derivation in R with the empty set of premisses and conclusion y. The set of derivable conclusions of R coincides with the least fixed point $\mu\hat{R}$ of $\hat{R}\colon \wp U \to \wp U$ defined by $\hat{R}(P) = \{y \in U \mid \text{ex. } X/y \in R \text{ s. t. } X \subseteq P\}$.

In logic programming terms, a rule $X/y \in R$ yields a Horn clause $y \leftarrow X$ [11]. The least fixed point $\mu\hat{R}$ coincides with minimal Herbrand model of the logic program corresponding to R and thus with the single stable model, as no negation is involved [11,16].

For expressing reachability and the satisfaction of knowledge formulæ in an epistemically guarded transition system $\Gamma = (S, E, L, S_0, \mathcal{T})$ over (P, A) as inductive rule systems, we use two types of judgements, one of the form $s \in^\Gamma S_\omega$ with $s \in S$ for "state s is reachable in Γ", and one of the form $s \models^\Gamma \varphi$ with $s \in S$ and $\varphi \in \Phi_{P,A}$ in negation normal form for "state s satisfies formula φ in Γ". The rules for reachability read:

$$\frac{}{s_0 \in^\Gamma S_\omega} \quad \text{if } s_0 \in S_0 \qquad \frac{s \in^\Gamma S_\omega}{s' \in^\Gamma S_\omega} \quad \begin{array}{l}\text{if ex. } (\varphi \supset B) \in \mathcal{T},\\ (s, s') \in B, \text{ and } s \models^\Gamma \varphi\end{array}$$

where $s \models^\Gamma \varphi$ in the side condition of the second rule requires this judgement to be derivable in the rule system for satisfaction. The rules for this system read:

$$\frac{}{s \models^\Gamma \text{true}} \text{ if } s \in^\Gamma S_\omega \qquad \frac{}{s \models^\Gamma p} \begin{array}{l}\text{if } s \in^\Gamma S_\omega,\\ p \in L(s)\end{array} \qquad \frac{}{s \models^\Gamma \neg p} \begin{array}{l}\text{if } s \in^\Gamma S_\omega,\\ p \notin L(s)\end{array}$$

$$\frac{s \models^\Gamma \varphi_1 \quad s \models^\Gamma \varphi_2}{s \models^\Gamma \varphi_1 \wedge \varphi_2} \qquad \frac{s \models^\Gamma \varphi_1}{s \models^\Gamma \varphi_1 \vee \varphi_2} \qquad \frac{s \models^\Gamma \varphi_2}{s \models^\Gamma \varphi_1 \vee \varphi_2}$$

$$\frac{s' \models^\Gamma \varphi}{s \models^\Gamma \mathsf{M}_a \varphi} \begin{array}{l}\text{if } (s,s') \in E_a,\\ s' \in^\Gamma S_\omega\end{array} \qquad \frac{(s' \models_\Gamma \varphi)_{s' \in^\Gamma S_\omega,\, (s,s') \in E_a}}{s \models^\Gamma \mathsf{K}_a \varphi}$$

Here, the last rule for satisfaction in fact is not monotone w.r.t. reachability: In order to infer $s \models^\Gamma \mathsf{K}_a \varphi$ it is not necessary to infer $s' \models^\Gamma \varphi$ for all s' with $(s,s') \in E_a$, but only for those for which $s' \in^\Gamma S_\omega$ can be deduced — and also for all of those.

The notion of providing epistemic witnesses allows to stratify the inductive rule systems according to the involved depth $k \geq 0$: We specialise the judgement $s \in^\Gamma S_\omega$ into $s \in^\Gamma S_k$ meaning "state s is reachable in Γ in up to k steps" and, similarly, the judgement $s \models^\Gamma \varphi$ into $s \models^\Gamma_k \varphi$ meaning "formula φ is satisfied in Γ at state s considering states reachable in up to k steps". The rules for reachability become for all $k \geq 0$:

$$\frac{}{s_0 \in^\Gamma S_k} \text{ if } s_0 \in^\Gamma S_0 \qquad \frac{s \in^\Gamma S_k}{s' \in^\Gamma S_{k+1}} \begin{array}{l}\text{if ex. } (\varphi \supset B) \in \mathcal{T},\\ (s,s') \in B, \text{ and } s \models^\Gamma_k \varphi\end{array}$$

Analogously the rules for satisfaction become for all $k \geq 0$:

$$\frac{}{s \models^\Gamma_k \text{true}} \text{ if } s \in^\Gamma S_k \qquad \frac{}{s \models^\Gamma_k p} \begin{array}{l}\text{if } s \in^\Gamma S_k,\\ p \in L(s)\end{array} \qquad \frac{}{s \models^\Gamma_k \neg p} \begin{array}{l}\text{if } s \in^\Gamma S_k,\\ p \notin L(s)\end{array}$$

$$\frac{s \models^\Gamma_k \varphi_1 \quad s \models^\Gamma_k \varphi_2}{s \models^\Gamma_k \varphi_1 \wedge \varphi_2} \qquad \frac{s \models^\Gamma_k \varphi_1}{s \models^\Gamma_k \varphi_1 \vee \varphi_2} \qquad \frac{s \models^\Gamma_k \varphi_2}{s \models^\Gamma_k \varphi_1 \vee \varphi_2}$$

$$\frac{s' \models^\Gamma_k \varphi}{s \models^\Gamma_k \mathsf{M}_a \varphi} \begin{array}{l}\text{if } (s,s') \in E_a,\\ s' \in^\Gamma S_k\end{array} \qquad \frac{(s' \models^\Gamma_k \varphi)_{s' \in^\Gamma S_k,\, (s,s') \in E_a}}{s \models^\Gamma_k \mathsf{K}_a \varphi}$$

In particular, the rules for $s \models^\Gamma_k \mathsf{M}_a \varphi$ and $s \models^\Gamma_k \mathsf{K}_a \varphi$ are sound for epistemically guarded transition systems providing epistemic witnesses. The notion of "providing epistemic witnesses" requires that, if $\mathsf{K}_a \varphi$ does not hold at depth k, there is a counterexample to φ at depth $\leq k$. The general case can be covered by dropping the depths and taking into account that $\mathsf{K}_a \varphi$ does not hold at some state s if, and only if, there is some reachable, a-indistinguishable state s' at which φ does not hold. Therefore, in order to derive that $\mathsf{K}_a \varphi$ indeed holds at some reachable state s, it is necessary and sufficient to show that it is *not* possible to derive that $\neg\varphi$ holds at some reachable, a-indistinguishable state s'.

5.2 General Rule Systems with Positive and Negative Premisses

For expressing negative information in terms of a rule system, we complement the positive premisses of the rules by negative ones: We consider general *rule systems* R over

a universe U consisting of rules of the form $(X, \not\vdash Z)/y$ where $X, Z \subseteq U$ are the *positive* and *negative premisses*, and $y \in U$ is the *conclusion*; it is interpreted as "if all X can be inferred but no Z, then y can be inferred". The *derivations* in R together with their *sets of positive and negative premisses* and *conclusions* are again inductively defined as follows:

- a $y \in U$ is itself a derivation; its set of positive premisses is $\{y\}$, its set of negative premisses is $\emptyset$, and its conclusion is y;
- if $(X, \not\vdash Z)/y \in R$ and $(d_x)_{x \in X}$ a family of derivations with conclusions $(x)_{x \in X}$, then $((d_x)_{x \in X}, \not\vdash Z)/y$ is a derivation; its set of positive premisses is the union of the positive premisses of $(d_x)_{x \in X}$, its set of negative premisses is the union of the negative premisses of $(d_x)_{x \in X}$ together with Z, and its conclusion is y.

For a $B \subseteq U$, let $\bar{R}(B)$ be all those $y \in U$ such that there is a derivation of y in R with the empty set of positive premisses and no negative premisses in B. The set of *derivable conclusions* of R is given by the least fixed point of $\bar{R}$ if it exists.

From the logic programming perspective, a general rule $(X, \not\vdash Z)/y \in R$ can be seen as a clause of the form $y \leftarrow X, \not\vdash Z$ with $\not\vdash$ read as "negation as failure" [5,11]. Checking that a $B \subseteq U$ is a "stable model" of the logic program obtained from R in this way corresponds to the following process on general rule systems: first the reduct R_B is formed by disregarding all rules $(X, \not\vdash Z)/y \in R$ with $B \cap Z \neq \emptyset$ and transforming the remaining rules $(X, \not\vdash Z)/y \in R$ into $X/y \in R_B$; then R_B is an inductive rule system and B is stable if $B = \mu\hat{R}_B$. In particular, the stable models correspond to the *solutions* of $\bar{R}(B) = B$.

With this generalised notion of rule systems we can reformulate and combine the two inference systems for reachability and satisfaction in an epistemically guarded transition system $\Gamma = (S, E, L, S_0, \mathcal{T})$ over (P, A) by using a single judgement $s \models^\Gamma_\omega \varphi$ for "state s satisfies φ in Γ and state s is reachable in Γ". A negative premiss $\not\vdash(s \models^\Gamma_\omega \mathrm{true})$ thus stands for "$s \in^\Gamma S_\omega$ cannot be deduced". The new rules with also negative premisses read:

$$\frac{}{s_0 \models^\Gamma_\omega \mathrm{true}} \text{ if } s_0 \in S_0 \qquad \frac{s \models^\Gamma_\omega \varphi}{s' \models^\Gamma_\omega \mathrm{true}} \text{ if ex. } (\varphi \supset B) \in \mathcal{T},\ (s, s') \in B$$

$$\frac{s \models^\Gamma_\omega \mathrm{true}}{s \models^\Gamma_\omega p} \text{ if } p \in L(s) \qquad \frac{s \models^\Gamma_\omega \mathrm{true}}{s \models^\Gamma_\omega \neg p} \text{ if } p \notin L(s)$$

$$\frac{s \models^\Gamma_\omega \varphi_1 \quad s \models^\Gamma_\omega \varphi_2}{s \models^\Gamma_\omega \varphi_1 \wedge \varphi_2} \qquad \frac{s \models^\Gamma_\omega \varphi_1}{s \models^\Gamma_\omega \varphi_1 \vee \varphi_2} \qquad \frac{s \models^\Gamma_\omega \varphi_2}{s \models^\Gamma_\omega \varphi_1 \vee \varphi_2}$$

$$\frac{s' \models^\Gamma_\omega \varphi}{s \models^\Gamma_\omega \mathsf{M}_a \varphi} \text{ if } (s, s') \in E_a \qquad \frac{s \models^\Gamma_\omega \mathrm{true} \quad \not\vdash(s' \models^\Gamma_\omega \mathrm{nnf}(\neg\varphi))_{(s,s') \in E_a}}{s \models^\Gamma_\omega \mathsf{K}_a \varphi}$$

The rule for $s \models^\Gamma_\omega \mathsf{K}_a \varphi$ checks that s is reachable, but that no counterexample to φ can be reached at an a-undistinguishable state.

Using general rule systems, the solvability of an epistemically guarded transition system is shifted to computing derivable conclusions. As for knowledge-based programs, it is not obvious from just the rules of a system R whether there are solutions of $\bar{R}(B) = B$ at all, and whether there is a least one.

Example 9. (a) The general rule system

$$R_0 = \left\{ \frac{x_1}{x_1}, \frac{\not\in x_1 \ \not\in x_2}{x_2} \right\} \quad \text{over} \quad \{x_1, x_2\}$$

has no set of derivable conclusions, since $\bar{R}_0$ has no fixed point; in particular, $\bar{R}_0(\emptyset) = \{x_2\}$ and $\bar{R}_0(\{x_1\}) = \emptyset = \bar{R}_0(\{x_2\})$. In terms of stable models, computing $\bar{R}_0(\emptyset)$ amounts to removing the negative premisses from the rule $(\emptyset, \not\in\{x_1, x_2\})/x_2$, such that the inductive rules $\{x_1\}/x_1$ and $\emptyset/x_2$ remain; and computing $\bar{R}_0(\{x_i\})$ leads to the single inductive rule $\{x_1\}/x_1$ for $i \in \{1, 2\}$.

R_0 also demonstrates that the set of derivable conclusions of a general rule system R need not coincide with the least fixed point of the operator $\hat{R}\colon \wp U \to \wp U$ when transferred from inductive rule systems by now setting $\hat{R}(P) = \{y \in U \mid \text{ex. } (X, \not\in Z)/y \in R \text{ s.t. } X \subseteq P, P \cap Z = \emptyset\}$: $\mu\hat{R}_0 = \{x_1\}$.

On the other hand, in view of the general rule system for epistemically guarded transition systems R_0 can also be rephrased as a knowledge-based program with a single agent a and a single variable $\mathrm{x} \in \{0, 1, 2\}$, which a cannot observe, started with $\mathrm{x} = 0$:

if $\mathsf{M}_\mathrm{a}\,\mathrm{x} = 1 \twoheadrightarrow \mathrm{x} \leftarrow 1$
$[\!]\ \mathsf{K}_\mathrm{a}(\mathrm{x} \neq 1 \wedge \mathrm{x} \neq 2) \twoheadrightarrow \mathrm{x} \leftarrow 2$ **fi**

(b) There may be several solutions of a general rule system, but no least one:

$$R_1 = \left\{ \frac{\not\in x_1}{x_3}, \frac{\not\in x_3}{x_1} \right\} \quad \text{over} \quad \{x_1, x_3\}$$

has the solutions $\{x_1\}$ and $\{x_3\}$, but $\emptyset$ is no solution. It corresponds to the "variable setting" knowledge-based program of the introduction, see Ex. 1(b):

if $\mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 1 \twoheadrightarrow \mathrm{x} \leftarrow 3$
$[\!]\ \mathsf{K}_\mathrm{a}\,\mathrm{x} \neq 3 \twoheadrightarrow \mathrm{x} \leftarrow 1$ **fi**

(c) Combining a contradictory rule $(\emptyset, \not\in\{x_1, x_2\})/x_2$ with the non-determined rules of R_1 we obtain the rule system

$$R_2 = \left\{ \frac{\not\in x_1}{x_3}, \frac{\not\in x_3}{x_1}, \frac{\not\in x_1 \ \not\in x_2}{x_2} \right\} \quad \text{over} \quad \{x_1, x_2, x_3\}$$

which has the unique solution $\{x_1\}$: if x_3 were inferable, i.e., x_1 not inferable, this would trigger the contradictory rule for x_2 (see Ex. 4(c)). □

5.3 Solving General Rule Systems

The observations and definitions for epistemic must/can transition structures and constructive interpretation, see Sect. 4.2, can now readily be transferred to a more abstract account for general rule systems. In fact, this reconstructs the "Kripke-Kleene fixpoint"

using under- and over-approximations [11], though now using an inductive partial order. We also relate the case where the constructive interpretation is not only monotone, but continuous to knowledge-based programs.

Define, for a universe U, the set $\wp^{\pm}U$ as $\{(P,Q) \in \wp U \times \wp U \mid P \subseteq Q\}$ and the relation $\subseteq^{\pm} \subseteq \wp^{\pm}U \times \wp^{\pm}U$ as $(P,Q) \subseteq^{\pm} (P',Q')$ if, and only if, $P \subseteq P'$ and $Q \supseteq Q'$.

Lemma 6. *$(\wp^{\pm}U, \subseteq^{\pm}, \bot^{\pm}_U)$ with $\bot^{\pm}_U = (\emptyset, U)$ is an inductive partial order.*

For a general rule system R over U with positive and negative premisses define the operator $\check{R}\colon \wp^{\pm}U \to \wp^{\pm}U$ that describes what *must* and what *can* be derived given what is assumed to be definitely and potentially derivable:

$$\begin{aligned}\check{R}(P,Q) = (&\{y \in U \mid \text{ex. } (X, \nprec Z)/y \in R \text{ s.t. } X \subseteq P,\ Q \cap Z = \emptyset\},\\ &\{y \in U \mid \text{ex. } (X, \nprec Z)/y \in R \text{ s.t. } X \subseteq Q,\ P \cap Z = \emptyset\})\end{aligned}$$

This is well-defined: if $(P,Q) \in \wp^{\pm}U$, then $\check{R}(P,Q) \in \wp^{\pm}U$, since for $P \subseteq Q$ and each $(X, \nprec Z)/y \in R$ with $X \subseteq P$ and $Q \cap Z = \emptyset$ it holds that $X \subseteq Q$ and $P \cap Z = \emptyset$. The operator is always monotone:

Lemma 7. *Let R be a rule system over U. If $(P_1,Q_1) \subseteq^{\pm} (P_2,Q_2)$, then $\check{R}(P_1,Q_1) \subseteq^{\pm} \check{R}(P_2,Q_2)$.*

As for constructive interpretation, Pataraia's fixed-point theorem now guarantees that the monotone operator $\check{R}$ on the inductive partial order $(\wp^{\pm}U, \subseteq^{\pm}, \bot^{\pm}_U)$ has a least fixed point. Again, it can be "computed" by possibly transfinite iterated application of $\check{R}$ to $\bot^{\pm}_U$. If, however, $\check{R}$ is even continuous, then, by Kleene's fixed-point theorem, it suffices to consider all finite approximations, i.e., $\mu\check{R} = \bigcup^{\pm}_{n\in\mathbb{N}} \check{R}^n(\bot^{\pm}_U)$; that $\check{R}$ is *continuous* means that if $\Delta \subseteq \wp^{\pm}U$ is directed, then $\bigcup^{\pm} \check{R}(\Delta) = \check{R}(\bigcup^{\pm} \Delta)$.

Lemma 8. *Let R be a rule system over U such that every rule of R has only finitely many positive and negative premisses. Then $\check{R}$ is continuous.*

The rule system for an epistemically guarded transition system $\Gamma = (S, E, L, S_0, \mathcal{T})$ over (P, A) always has only finitely many positive premisses; if for each $s \in S$ and each $a \in A$ the set $\{s' \in S \mid (s,s') \in E_a\}$ is finite, then there are also only finitely many negative premisses, such that the corresponding must/can operator is continuous.

6 Reasoning About Knowledge-based Programs

We have implemented the constructive interpretation of knowledge-based programs in the prototypical "Temporal Epistemic Model Interpreter and Checker" (TEMIC[5]). The tool first computes the least constructive fixed point of a (finite state) epistemically guarded transition system. If the least fixed point is decided, the least solution in terms of epistemic transition structures has been found; otherwise it is checked whether the re-interpretation using the lower bound of the undecided least fixed point yields a solution.

[5] https://bitbucket.org/knappale/temic

If either succeeds, properties of the resulting model can be checked. These properties can be expressed in CTLK, the combination of the branching "Computation Tree Logic" (CTL) and epistemic logic [21]. What is more, CTLK can also be used in tEmIc for the action guards. The constructive interpretation just evaluates each universal quantifier of a CTL formula — A for "on all paths" — over the upper bound and each existential quantifier — E for "on some path" — over the lower bound. This adds the temporal dimension to the domain of application of knowledge-based programs. For the run-based interpreted systems of Fagin et al. [13], Van der Hoek and Woolridge [20] and Su [27] provide transformations for linear-time model checking based on local propositions, though for a fixed set of runs that does not depend on the evaluation of knowledge guards. The CTLK-model checker MCMAS [21] similarly operates on a fixed, predetermined model. In dynamic epistemic logic and its model checker DEMO [31], the transition structure is given by epistemic actions.

We first recapitulate briefly CTLK and then show its constructive evaluation over epistemic must/can transition structures. We next describe tEmIc by means of the bit transmission problem and the small paradoxical exercise of the "unexpected examination"; the tEmIc distribution also contains specifications for the well-known problems "Muddy Children" [31, pp. 93ff.] and "Sum-and-Product" [31, pp. 96f.]. Finally, we proceed to an application where CTLK is also used in the action guards: the Java memory model.

6.1 CTLK

The *CTLK-formulæ* over (P, A) are defined by the following grammar:

$$\varphi ::= p \mid \text{false} \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathsf{K}_a\,\varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{EG}\,\varphi \mid \mathsf{E}[\varphi_1\,\mathsf{U}\,\varphi_2]$$

where $p \in P$ and $a \in A$. The path quantifier E is interpreted as "there is a path", the temporal modality X as "in the next step", G as "always", and U as "until". We also consider the path quantifier A for "on all paths" and the modalities F for "eventually" and R for "release", such that $\neg\mathsf{EG}\,\neg\varphi$ is abbreviated by $\mathsf{AF}\,\varphi$ and $\neg\mathsf{E}[\neg\varphi_1\,\mathsf{U}\,\neg\varphi_2]$ by $\mathsf{A}[\varphi_1\,\mathsf{R}\,\varphi_2]$. The *satisfaction relation* $M, s \models \varphi$ of a CTLK-formula φ over (P, A) at state $s \in S$ of an epistemic transition structure $M = (S, E, L, S_0, T)$ over (P, A) conservatively extends the satisfaction relation of epistemic formulæ by

$$\begin{aligned}
M, s \models \mathsf{EX}\,\varphi &\iff \text{ex. } s_0, s_1, \ldots \in \mathscr{P}(M, s) \text{ s.t. } M, s_1 \models \varphi \\
M, s \models \mathsf{EG}\,\varphi &\iff \text{ex. } s_0, s_1, \ldots \in \mathscr{P}(M, s) \text{ s.t. } M, s_i \models \varphi \text{ f.a. } i \in \mathbb{N} \\
M, s \models \mathsf{E}[\varphi_1\,\mathsf{U}\,\varphi_2] &\iff \text{ex. } s_0, s_1, \ldots \in \mathscr{P}(M, s) \text{ and } l \in \mathbb{N} \text{ s.t.} \\
&\qquad M, s_i \models \varphi_1 \text{ f.a. } 0 \leq i < l \text{ and } M, s_l \models \varphi_2
\end{aligned}$$

where $\mathscr{P}(M, s)$ denotes all *paths* of M, i.e., the infinite state sequences $s_0, s_1, \ldots \in S$ with $s_0 = s$ and $(s_i, s_{i+1}) \in T$ for all $i \in \mathbb{N}$. A CTLK-formula φ is *valid* in M, written $M \models \varphi$, if it is satisfied in all initial states, i.e., $M, s_0 \models \varphi$ for all $s_0 \in S_0(M)$.

For a direct definition of the satisfaction of CTLK-formulæ with an A, the existential path quantification for E has to be replaced by universal path quantification. As for simple epistemic logic, CTLK including $\mathsf{AX}\,\varphi$, $\mathsf{AG}\,\varphi$ etc. admits a negation normal form (see, e.g., [3, pp. 333f.]). The *constructive satisfaction relation* of a CTLK-formula in negation

normal form over an epistemic must/can transition structure $Y = (S, E, L, S_0, \mathcal{T})$ over (P, A) at a state $s \in S_\omega(Y_\nu)$, written $Y, s \models \varphi$, conservatively extends the constructive satisfaction relation of epistemic formulæ and interprets E over the lower bound Y_μ and A over the upper bound Y_ν such that, in particular,

$$Y, s \models \mathsf{EF}\,\varphi \iff \text{ex. } s_0, s_1, \ldots \in \mathscr{P}(Y_\mu, s) \text{ and } i \in \mathbb{N} \text{ s. t. } Y, s_i \models \varphi$$
$$Y, s \models \mathsf{AF}\,\varphi \iff \text{f. a. } s_0, s_1, \ldots \in \mathscr{P}(Y_\nu, s) \text{ ex. } i \in \mathbb{N} \text{ s. t. } Y, s_i \models \varphi$$

6.2 tEmIc

tEmIc is a symbolic model interpreter and checker for epistemically guarded transition systems using CTLK. It is written in Java and uses binary decision diagrams for state space representation [28]; it also supports bounded integers and their arithmetic. Given a specification, tEmIc first computes the least constructive fixed point by iterated must/can interpretation. If this fixed point is not decided it checks whether another interpretation using the lower bound of the fixed point yields a solution. If either succeeds, tEmIc proceeds with model checking given properties; these statements can be specified as CTLK-formulæ which have to hold in all initial states or as a reachability query. Reachable deadlock states without outgoing transitions result in a warning.

For example, the bit transmission problem of the introduction as formalised in Ex. 1(a) can be represented as a tEmIc specification as follows (rules are introduced by keyword `action` followed by a name of the rule and the rule definition):

```
var sbit, ack, rbit, snt : boolean initial (ack | rbit | snt) <-> false;

agent S = { sbit, ack };  agent R = { rbit, snt };
let R_knows_bit = exists bit:boolean . K[R] sbit <-> bit;

action S_sends_bit_ok
guard not K[S] R_knows_bit do rbit := sbit, snt := true;
action S_sends_bit_failed
guard not K[S] R_knows_bit do ;
action R_sends_ack_ok
guard R_knows_bit and not K[R] K[S] R_knows_bit do ack := true;
action R_sends_ack_failed
guard R_knows_bit and not K[R] K[S] R_knows_bit do ;
```

Constructive interpretation yields in a few milliseconds the decided least fixed point of Ex. 2, over which some CTLK-properties can be checked:

```
check initial EF           R_knows_bit;
check initial EF      K[S] R_knows_bit;
check initial EF K[R] K[S] R_knows_bit;
```

The first two are reported to hold, but the last does not since agent R cannot gather enough information to be sure that the bit has been received by agent S.

For another example, consider the "unexpected examination" paradox [10, Sect. 4.7, there called "unexpected hanging"] (for a detailed account see, e. g., [26, Sects. 5.2f.]): A class is told that within the next week there will be an exam, but it will be a surprise. The class might reason that the exam cannot happen on Friday, because if there has been no exam up to Thursday it will not be a surprise on Friday any more; by backward induction it might reason that there cannot be a surprise exam in the next week at all. This problem statement can be readily expressed as a tEmIc specification:

```
var day : 0..5 initial day = 0;
var exam : 0..4;
var written : boolean initial written <-> false;

agent P = { day, written };

action act1
guard day < 5 and (day = exam) and (not K[P] day = exam) and not written
do written := true, day := day+1;
action act2
guard day < 5 and (day != exam) do day := day+1;
action stutter
do ;
```

Again, constructive interpretation yields in a few milliseconds a decided least fixed point. Over this epistemic transition structure we can check that on, e. g., Wednesday the exam can be written and still is indeed a surprise:

```
check reachable exam = 2 & written;
```

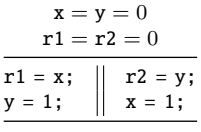
For such a reachability check tEmIc also provides a witness that tells that `act2` is executed twice after which `act1` follows. The following CTLK-property, however, is not satisfied, as it would have to hold in all initial states — and with `exam` being 4 the class cannot be surprised any more:

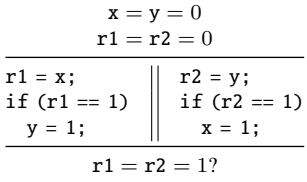
```
check initial EF written;
```

6.3 Memory Models

Memory models regulate the interaction between threads, their caches, and the main memory [23]. The original Java memory model — one of the first formal such models — has been harshly criticised for making several compiler optimisations impossible and has subsequently been superseded by a more liberal model [17, Ch. 17]. Keeping strong guarantees for sequentially consistent, well-synchronised programs, reorderings of data-independent statements or early, "prescient" reads from other threads are allowed for programs with data races. Still, some limits, like consistency with data or control flow dependencies or no "out-of-thin-air" values, should be in force [25,2].

For example, in the following two-threaded Java-like program to the left it should be possible that both thread-local registers `r1` and `r2` are assigned the value 1 when reading the global, shared variables `x` and `y`: A compiler could reorder the data-independent statements in both threads. This behaviour, however, should be forbidden in the example to the right, since there is a symmetric data dependence.





We want to capture the behaviour of a multi-threaded (Java) program with a liberal memory model without having to check all possible compiler transformations — the

correctness of such transformations would actually depend on the program semantics including the memory model. In fact, in the current Java memory model out-of-order executions have to be justified by other legal executions. We interpret these justifications as witnesses in terms of knowledge-based programs; our current exposition, however, neglects synchronisation. We first represent the state space of a two-threaded (Java) program like the ones above by the following tEmIc declarations:

```
var x, y, r1, r2 : 0..2 initial x = 0 & y = 0 & r1 = 0 & r2 = 0;
var step1, step2 : 1..3 initial step1 = 1 & step2 = 1;

agent t1 = { step1, r1 };  agent t2 = { step2, r2 };
```

The thread agents `t1` and `t2` can only observe their local registers and their program counters. The program steps for both threads are turned into actions like

```
action t1_1 guard step1 = 1 do r1 := x, step1 := step1+1;
action t1_2 guard step1 = 2 do y := 1, step1 := step1+1;
```

Additionally, we allow for a "prescient reading" of the value v from the main memory variable x by thread θ into the local variable r at step s by the following action:

```
action readθ_x_v_r_s
guard stepθ = s and K[θ] (EF (r = 0 & x = v) and EF (r = v & x = v))
do r := v, stepθ := stepθ+1;
```

The thread θ can read v from x into r early on if it *knows* that *there is an execution* where x has value v without dependence on already setting r to v, and, furthermore, that *there is an execution* where the early setting is confirmed. The statement `r1 = x;` of the first thread is expanded into three read actions `read1_x_0_r1_1`, `read1_x_1_r1_1`, and `read1_x_2_r1_1` plus the plain reading action `t1_1`. With this encoding, tEmIc reports that for the first example to the left it is indeed possible to obtain $\texttt{r1} = \texttt{r2} = 1$ in the least constructive fixed point, but that this is impossible for the example to the right.

A more intriguing case is presented by the following two examples: According to Manson et al. [23, pp. 35f.] (cf. also [2]), the program to the left can result in $\texttt{r1} = \texttt{r2} = \texttt{r3} = 1$:

$\texttt{x} = \texttt{y} = 0$, $\texttt{r1} = \texttt{r2} = \texttt{r3} = 0$

`r1 = x;`		`r3 = y;`
`if (r1 == 0)`		`x = r3;`
`  x = 1;`		
`r2 = x;`		
`y = r2;`		

$\texttt{r1} = \texttt{r2} = \texttt{r3} = 1?$

$\texttt{x} = \texttt{y} = 0$, $\texttt{r1} = \texttt{r2} = \texttt{r3} = 0$

`r1 = x;`		`r2 = x;`	`r3 = y;`
`if (r1 == 0)`		`y = r2;`	`x = r3;`
`  x = 1;`			

$\texttt{r1} = \texttt{r2} = \texttt{r3} = 1?$

A compiler could see that only 0 and 1 are possible for `x` and `y` and "can then replace `r2 = x` by `r2 = 1`, because either 1 was read from `x` on line 1 and there is no intervening write, or 0 was read from `x` on line 1, 1 was assigned to `x` on line 3, and there was no intervening write"; this definite assignment can be used to transform the last line to `y = 1;` which finally can be made the first action of the first thread, as there are no dependencies. But the same transformation is not possible for the program to the right, and there the same behaviour should be disallowed. Still, the left program is the result of inlining the second thread into the first. Our encoding of the two programs in tEmIc

confirms these considerations and the witness for the left program indeed first sets `r3` to 1 and confirms this only in the last step setting `y` to 1.

7 Conclusions and Future Work

We have introduced a must/can analysis for the interpretation of knowledge-based programs inspired by the constructive semantics of synchronous programming languages. The resulting constructive interpretation provides lower and upper bounds for the possible executions. This interpretation has been shown to be monotone and to yield a least fixed point. We have also transformed knowledge-based programs to general rule systems with positive and negative premisses. Finally, we have described our tool tEmIc for constructive interpretation and temporal-epistemic model checking over CTLK and demonstrated some applications of interpreting knowledge-based programs including CTLK-guards.

Our epistemic logic could be complemented by group knowledge [14, Ch. 6], like common or distributed knowledge. The temporal dimension could be extended to "Linear-Time Logic" (LTL), and, more importantly, to include some notion of fairness. Criteria for ensuring decided least fixed points for the must/can interpretation beyond synchronicity would be desirable. Also a comparison with non-monotone inductive definitions [12], SOS rules with negative premisses [24], and solution strategies for epistemic specifications [5], would be of interest. On the other hand, the general constructive approach may be useful to complement existing intuitionistic approaches to the semantics of synchronous programming languages [22]. Finally, the domain of memory models should be covered more comprehensively by interpreting knowledge-based programs.

References

1. Aczel, P.: An introduction to inductive definitions. In: Barwise, J. (ed.) Handbook of Mathematical Logic, chap. C.7, pp. 783–818. North-Holland (1977)
2. Aspinall, D., Ševčík, J.: Java Memory Model examples: Good, bad and ugly. In: Proc. Verification and Analysis of Multi-Threaded Java-like Programs (VAMP 2007) (2007)
3. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
4. Baltag, A., Moss, L.S.: Logics for Epistemic Programs. Synth. **139**(2), 165–224 (2004). https://doi.org/10.1023/B:SYNT.0000024912.56773.5e
5. Baral, C., Gelfond, M.: Logic programming and knowledge representation. J. Logic Program. **19–20**(Suppl. 1), 73–148 (1994)
6. Benveniste, A., Caspi, P., Edwards, S.A., Halbwachs, N., Guernic, P.L., de Simone, R.: The synchronous languages twelve years later. Proc. IEEE **91**(1), 64–83 (2003)
7. Berry, G.: The foundations of Esterel. In: Plotkin, G., Stirling, C., Tofte, M. (eds.) Proof, Language and Interaction: Essays in Honour of Robin Milner, pp. 425–454. Foundations of Computing Series, MIT Press (2000)
8. Berry, G.: The Constructive Semantics of Pure Esterel, Draft v3 (2002), https://www-sop.inria.fr/members/Gerard.Berry/Papers/EsterelConstructiveBook.pdf
9. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order. Cambridge University Press, 2nd edn. (2002)
10. de Haan, H.W., Hesselink, W.H., de Lavalette, G.R.R.: Knowledge-based asynchronous programming. Fund. Inform. **63**(2-3), 259–281 (2004)

11. Denecker, M., Bruynooghe, M., Marek, V.: Logic programming revisited: Logic programs as inductive definitions. ACM Trans. Comput. Logic **2**(4), 623–654 (2001)
12. Denecker, M., Ternovska, E.: A logic of nonmonotone inductive definitions. ACM Trans. Comput. Log. **9**(2), 14:1–14:52 (2008)
13. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Knowledge-based programs. Distr. Comput. **10**(4), 199–225 (1997)
14. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. MIT Press (2003)
15. Fandinno, J., Faber, W., Gelfond, M.: Thirty years of epistemic specifications. Theo. Pract. Logic Program. **22**(6), 1043–1083 (2022)
16. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Kowalski, R.A., Bowen, K.A. (eds.) Proc. 5th Intl. Conf. Symp. Logic Programming. pp. 1070–1080. MIT Press (1988)
17. Gosling, J., Joy, B., Steele, G., Bracha, G.: The Java Language Specification. Addison-Wesley, 3rd edn. (2005)
18. Halbwachs, N., Caspi, P., Raymond, P., Pilaud, D.: The synchronous data-flow programming language Lustre. Proc. IEEE **79**(9), 1305–1320 (1991)
19. Harper, R.: Practical Foundations of Programming Languages. Cambridge University Press (2013)
20. van der Hoek, W., Wooldridge, M.J.: Model checking knowledge and time. In: Bosnacki, D., Leue, S. (eds.) Proc. 9th Intl. Ws. Model Checking of Software (SPIN 2002). Lect. Notes Comp. Sci., vol. 2318, pp. 95–111. Springer (2002). https://doi.org/10.1007/3-540-46017-9_9
21. Lomuscio, A., Penczek, W.: Model checking temporal epistemic logic. In: van Ditmarsch et al. [29], chap. 8, pp. 397–441
22. Lüttgen, G., Mendler, M.: The intuitionism behind statecharts steps. ACM Trans. Comput. Log. **3**(1), 1–41 (2002). https://doi.org/10.1145/504077.504078
23. Manson, J., Pugh, W., Adve, S.: The Java memory model (2005), http://dl.dropbox.com/u/1011627/journal.pdf, draft
24. Mousavi, M., Phillips, I., Reniers, M.A., Ulidowski, I.: Semantics and expressiveness of ordered SOS. Inform. & Comput. **207**(2), 85–119 (2009). https://doi.org/10.1016/j.ic.2007.11.008
25. Pugh, W.: The Java memory model (1999–), http://www.cs.umd.edu/~pugh/java/memoryModel/
26. Sainsbury, R.M.: Paradoxes. Cambridge University Press, 3rd edn. (2009)
27. Su, K.: Model checking temporal logics of knowledge in distributed systems. In: McGuiness, D.L., Ferguson, G. (eds.) Proc. 19th Natl. Conf. Artificial Intelligence, 16th Conf. Innovative Applications of Artificial Intelligence (AAAI 2004). pp. 98–103. AAAI Press, MIT Press (2004)
28. Vahidi, A.: JDD: A pure Java BDD and Z-BDD library. https://bitbucket.org/vahidi/jdd (2003)
29. van Ditmarsch, H., Halpern, J.Y., van der Hoek, W., Kooi, B. (eds.): Handbook of Epistemic Logic. College Publ. (2015)
30. van Ditmarsch, H., Halpern, J.Y., van der Hoek, W., Kooi, B.: An introduction to logics of knowledge and belief. In: Handbook of Epistemic Logic [29], chap. 1, pp. 1–51
31. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic Epistemic Logic, Synthese Library, vol. 337. Springer (2008)